A COMPUTATIONAL ONTOLOGICAL MODEL FOR MACHINE-UNDERSTANDABLE DATA IN ARTIFICIAL INTELLIGENCE

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF SOCIAL SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DİLEK YARGAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
THE DEPARTMENT OF PHILOSOPHY

OCTOBER 2022

Approval of the thesis:

A COMPUTATIONAL ONTOLOGICAL MODEL FOR MACHINE-UNDERSTANDABLE DATA IN ARTIFICIAL INTELLIGENCE

submitted by **DİLEK YARGAN** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Philosophy, the Graduate School of Social Sciences of Middle East Technical University** by,

Prof. Dr. Sadettin KİRAZCI
Dean
Graduate School of Social Sciences _____

Prof. Dr. Halil Ş. TURAN
Head of Department
Department of Philosophy _____

Assoc. Prof. Dr. Aziz F. ZAMBAK
Supervisor
Department of Philosophy _____

Examining Committee Members:

Prof. Dr. Sevinç GÜLSEÇEN (Head of the Examining Committee)
İstanbul University
Department of Informatics _____

Assoc. Prof. Dr. Aziz F. ZAMBAK (Supervisor)
Middle East Technical University
Department of Philosophy _____

Prof. Dr. Erdoğan DOĞDU
Angelo State University
Department of Computer Science _____

Prof. Dr. Ayhan SOL
Middle East Technical University
Department of Philosophy _____

Prof. Dr. David GRÜNBERG
Middle East Technical University
Department of Philosophy _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Dilek YARGAN

Signature :

ABSTRACT

A COMPUTATIONAL ONTOLOGICAL MODEL FOR
MACHINE-UNDERSTANDABLE DATA IN ARTIFICIAL INTELLIGENCE

YARGAN, Dilek
Ph.D., Department of Philosophy
Supervisor: Assoc. Prof. Dr. Aziz F. ZAMBAK

October 2022, 383 pages

Big Data is believed to be one of the most important phenomena of the century due to its transformative effects and those undeniable effects of the data deluge in the fields of industry, science, and the Web –the fundamental wheels upon which today's world makes progress– are, however, hardly transformative. If Big Data is to lead to revolutions in these fields, the machine must be autonomous, that is, to perform higher-order cognitive skills, such as making decisions, inferences, and recommendations. The condition for an autonomous machine is its ability to *understand* and process Big Data. So, this work aims to determine the foundations for machine-understandability and accordingly determine the conditions for an autonomous machine. In this respect, this work proposes a machine ontology, Ontology 4.0, which represents phenomena in terms of their semantic properties in a relation-based fashion and processes semantic properties according to their types. Adapted from trope and type theories, this ontology forms the basis for machine-understandability. However, due to computational limitations from type theories, the semantic properties are discussed to be represented with nonextensional ontological objects, viz., *urtropes*, which turn out to be the building blocks of Ontology 4.0. Category theory is adapted as a formaliza-

tion tool to implement this structure in the machine. Consequently, providing basis for machine-understandability, Ontology 4.0 is a machine ontology that harmonizes urtrope and modified category theory.

Keywords: machine ontology, machine-understandability, urtrope theory, autonomous machine, data science

ÖZ

YAPAY ZEKADA MAKİNE-ANLAYABİLİR VERİ İÇİN BERİMSEL BİR ONTOLOJİK MODEL

YARGAN, Dilek
Doktora, Felsefe Bölümü
Tez Yöneticisi: Doç. Dr. Aziz F. Zambak

Ekim 2022 , 383 sayfa

Büyük Veri dönüştürücü etkileri nedeniyle yüzyılın en önemli fenomenlerinden biridir, ancak veri tufanının tartışılmaz etkileri günümüz dünyasının ilerleme kaydettiği temel çarklar olan endüstri, bilim ve Web söz konusu olduğunda henüz dönüştürücü etkiye sahip değildir. Büyük Veri'nin bu alanlarda devrimlere yol açabilmesi için, makinenin otonom olması, yani karar verme, çıkarım yapma ve tavsiye verme gibi üst düzey bilişsel becerileri gerçekleştirmesi gereklidir. Makinenin otonom olmasının koşulu ise Büyük Veri'yi *anlama* ve işleme yeteneğidir. Dolayısıyla bu çalışma, makine-anlayabilirliğinin temellerini ve buna göre de makinenin otonom olmasının koşullarını belirlemeyi amaçlamaktadır. Bu bağlamda, bu çalışma, fenomenleri anlamsal özellikleriyle ilişki temelli bir şekilde temsil eden ve anlamsal özellikleri tiplerine göre işleyen bir makine ontolojisi olan Ontoloji 4.0'ı önermektedir. Trop ve tip kuramlarından uyarlanan bu ontoloji, makine-anlayabilirliğinin temelini oluşturur. Bununla birlikte, bu çalışmada tip kuramlarından gelen hesaplama sınırlamaları nedeniyle, anlamsal özelliklerin, Ontoloji 4.0'ın yapı taşlarını oluşturacak uzamsal olmayan ontolojik nesneler, yani *urtroplar* ile temsil edilmesi gerekliliğini de tartışacaktır. Kate-

gori kuramı, bu yapıyı makinede uygulayabilecek en belgin biçimsel araç olarak uyarlanmıştır. Sonuç olarak, makine-anlayabilirliğinin temelini oluşturan Ontoloji 4.0, urtrop ve değiştirilmiş kategori kuramlarının uyumlaştığı bir makine ontolojisidir.

Anahtar Kelimeler: makine ontolojisi, makine-anlayabilirliği, urtrop kuramı, otonom makine, veri bilimi

To Başar Yargan, my uplifter

ACKNOWLEDGMENTS

I am grateful to Beauty for surrounding us with eudaimonia and ataraxia.

I want to express my most profound appreciation and respect to my supervisor, Aziz F. Zambak, for his unfailing support, invaluable guidance, and continuous encouragement throughout my Ph.D. degree. His exceptional knowledge and vision gave me a new life.

I would like to extend my sincere thanks to Sevinç Gülseçen, Vedat Kamer, and Özlem Özdemir, who never wavered in their support.

Getting through the long and painful process of completing my dissertation would not have been possible without Onur Eylül Kara, Tuğba Tural, Selen Sahillioğlu, Svitlana Nesterova Coşkun, Özge Keskin, Murat Ulubay, Berk Yaylım, and Mete Kurtoğlu. They are always with me, profoundly believing in my work and abilities.

I am also thankful to my dear big family Huriye Satıoğlu, Ayhan Satıoğlu, Merve Ayşe Satıoğlu, Melek Arıkök, Onur Arıkök, Ayhan Arıkök, Rıza Eren Arıkök, Güler Yargan, Faruk Yargan, Başak Yargan, and Arda Çağın Mescigil.

Last and foremost, I want to extend my deepest and heartiest gratitude to my loving husband, Başar Yargan, for his constant support, tremendous encouragement, invaluable patience, and great tenderness.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AEO	Anatomical Entity Ontology
BFO	Basic Formal Ontology
CCC	Cartesian Closed Category
CERN	European Organization for Nuclear Research
CPS	Cyber-Physical System
CSV File	Comma-Separated Values File
CT	Category Theory
DARPA	The Defence Advanced Research Projects Agency
DIKW	Data-Information-Knowledge-Wisdom
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
ER Model	Entity-Relation Model
FOAF	Friend of a Friend
GPT	Generative Pre-trained Transformer
HAI	Human–Agent Interaction
HCI	Human–Computer Interaction
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IRI	Internationalized Resource Identifier
ISW	Industry, Science, Web
IoT	Internet of Things
KIF	Knowledge Interchange Format
KR	Knowledge Representation
LSST	Large Synoptic Survey Telescope
MeSH	Medical Subject Headings
ML	Machine Learning
NIST	The National Institute of Standards and Technology
NLP	Natural Language Processing
OBO	Open Biological and Biomedical Ontology
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema

RIF	Rule Interchange Format
ST	Set Theory
SUMO	Suggested Upper Merged Ontology
SQL	Structured Query Language
SPARQL	SPARQL Protocol and RDF Query Language
TLP	Tractatus Logico-Philosophicus
TM	Turing Machine
ULO	Upper-Level Ontology
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language

CHAPTER 1

INTRODUCTION

The belief that Big Data is providing/will provide transformations in the course of human history has made it one of the most important phenomena of the century. However, when the three main fields –industry, science, and the Web– that shape human history are analyzed, it cannot be said there has been a genuine transformation in these fields. Since in order to talk about a transformation that Big Data can provide in these fields, it is necessary for the machine to be autonomous, that is, to make decisions, make inferences, and make recommendations. In other words, an autonomous machine means a machine that can understand and process Big Data. This dissertation aims to identify representational, namely, philosophical and computational principles of machine-understandability, explore an appropriate data processing structure paving the way for machine-understandability, investigate how data should be represented, and find a formalization to implement and process the agreed-on data representation on the machine.

This chapter will introduce the work by first discussing what Big Data is and the failure of its expected impact in the fields of industry, science, and the Web, followed by explaining that the real impact depends on creating an autonomous machine. Then, it will discuss what structured data an autonomous machine should process and how they should be formalized. Lastly, it will elucidate the significance and, finally, the limitations of this work.

1.1 Big Data, Data Science, Autonomous Machines

This part provides a broad overview of autonomous machines and the concepts of machine-understandability. So, this part gives a foundational understanding of the dissertation.

1.1.1 Small and Big Data

> Data! Data! Data! I can't make bricks without clay.
>
> — Sherlock Holmes

Data always come first.[1] It is the indispensable component of any cognitive task. As an engine cannot function without the appropriate energy source specific to its usage, any scientific task cannot be conveyed without enough data on the research topic. Or, a machine cannot compute anything if there is no data, even if it has all the possible algorithms to solve any problem. Yet, what is data in information systems?

Various data definitions are introduced from different perspectives. The first definition of data is from Ackoff (1989, p. 3): "Data are symbols that represent the properties of objects and events." Objects and events can be symbolized in various sorts: characters, numbers, audio and visual signals, images, and alike. This definition is prevalent among information scientists, but there are other popular definitions as well. Shannon (1948), a prominent information theorist, defines data in terms of entropy as patterns of physical symbols/signs. Besides several definitions of data, most of the time, it is defined in terms of information as being units/morsels/pieces of information (Cf. Gitelman, 2013).[2] Furthermore, Mealy (1967, p. 525) distinguishes three distinct realms in the field

[1] In this work, 'data' is often treated as a mass noun; when required, it is used in its singular or plural forms.

[2] This sort of approach is problematic since labeling something as data or information depends on the perspective. Consider a timetable. When it is used for extracting a piece of certain information, it is data. Since the table contains texts and numbers, some facts are based on records or observations. On the other hand, a timetable can be information since it is refined data based on analyses and used for making judgments on facts.

of data processing in his work on the foundations of data modeling: the real world itself, ideas about it existing in the minds of humans, and symbols on paper or some other storage medium. Data can be defined as fragments of a real-world theory, and data processing juggles representations of these fragments of theory. Frické (2015, p. 652) also offers a different perspective and introduces data with a semantic value and in a machine-oriented way: "Data is anything recordable in a relational database in a semantically and pragmatically sound way." Lastly, consider the perspective of Mayer-Schoenberger and Cukier (2013, p. 78), who state, "[t]o datafy a phenomenon is to put it in a quantified format so it can be tabulated and analyzed;" accordingly, data is a description of something that is "recorded, analyzed, and reorganized."

The word 'data' has become the most attractive term in our day. Popular articles and books take data as the most valuable resource, "new oil of the digital age" (Leonelli, 2014). What made data so trendy is the *data deluge*: the exponential growth of data. We have generated 2.5 quintillion bytes per day in 2017, and the number has increased dramatically since the global internet population has increased 36.84% from 2017 to July 2021.[3] We can think of our daily life and figure out the data sources we engage with, for instance, data from sensors, social media, transactional applications, web searches, networks, instant message services, and alike. Thus, the data generated and stored is growing in depth and width. That is the essential feature of data coined 'big:' here we have *Big Data*.

Like data, Big Data also has several definitions. One approach that focuses on the characteristics of data deluge defines Big Data in terms of some Vs: velocity, variety, volume, veracity, variability, visualization, and value (Van Rijmenam, 2013).[4] Mayer-Schoenberger and Cukier (2013), on the other hand, focus on the opportunities of Big Data, and define it as extracting "new insights or create new forms of value, in ways that change markets, organizations, the relationship between citizens and governments, and more." Floridi (2012) points out the ambiguous use of the term 'Big Data,' and he addresses its definition from the

[3] See https://www.domo.com/learn/infographic/data-never-sleeps-9 for up-to-date numbers.

[4] Laney, enriched by Marr firstly suggested the list of Vs, van Riejmenam puts it in the final form.

document of Natural Science Foundation (NSF-12-499):

> The phrase 'big data' in this solicitation refers to large, diverse, complex, longitudinal, and/or distributed data sets generated from instruments, sensors, Internet transactions, email, video, click streams, and/or all other digital sources available today and in the future.

Ward and Barker (2013, p. 2) have collated and analyzed definitions of Big Data. They stress three common factors in definitions of Big Data: size, complexity, and technologies. The final definition that they reach is as follows:

> Big data is a term describing the storage and analysis of large and or complex data sets using a series of techniques including, but not limited to: NoSQL, MapReduce and machine learning.

boyd and Crawford (2012, p. 663) introduce Big Data as "a cultural, technological, and scholarly phenomenon" that leads on the interaction between technology –for maximizing computational power-, analysis – for drawing economic, social, technical, and legal inferences, and mythology –for escaping the belief of the more data, the more insight. The last definition is from Kitchin (2013, p. 262) as a conclusion.

> [A] number of key features of [...] big data are
>
> - huge in *volume*, consisting of terabytes or petabytes of data;
> - high in *velocity*, being created in or near real-time;
> - diverse in *variety*, being structured and unstructured in nature;
> - *exhaustive* in scope, striving to capture entire populations or systems;
> - fine-grained in *resolution*, aiming to be as detailed as possible, and uniquely *indexical* in identification;
> - *relational* in nature, containing common fields that enable the conjoining of different data sets and
> - *flexible*, holding the traits of extensionality (can add new fields easily) and scalability (can expand in size rapidly).

According to these definitions, it is seen that the difference between data and Big Data is not only "size." The prominence and status of Big Data is its impact on all areas of life. Most of the time, Big Data is thought of as the source from which we can draw solutions to all of our problems/concerns/issues regardless of the field. It has been thought to be a powerful tool that would surely provide discoveries that even cannot be imagined by humankind (Cf. boyd & Crawford, 2012). À la Mayer-Schoenberger and Cukier (2013, p. 19), "Big Data is all about seeing

and understanding the relations within and among pieces of information that, until very recently, we struggled to fully grasp." Consequently, it has aroused the interest of people from business to politics, science to philosophy.

There is a great deal of trust in Big Data, as it helps us gain insight that we could not have with traditional scientific tools. As Rosling (2010, 37:23–37:30) states that the more data there is the more discoveries can be made, and Floridi (2012) argues that new patterns can be detected in data, which results in the advancement of knowledge, and that studying Big Data is very important in our era. It is often highlighted that there is no need for theories, hypotheses, models, or human expertise in the data deluge era. We even do not need to find any causation since the correlations are proficient in providing "results." One of the most famous articles in this line of thought is of Anderson (2008), the then editor of Wired, whose article title lays bare his opinion on Big Data, "The end of theory: The data deluge makes the scientific method obsolete." According to him, the desired outcomes emerge when better data meet better analytical tools. The success of Google, for instance, lies merely in statistical methods, which demand neither semantic nor causal analyses. The scientific methods –constructing hypotheses and models, applying the data into models, and verifying the hypotheses- are obsolete. That statistical algorithms find the patterns where science cannot since correlations are enough for this task (Anderson, 2008; Mayer-Schoenberger & Cukier, 2013), and it is legitimate to analyze data without a hypothesis in data science. Prensky (2009) belittles even educated guesses on understanding the world by constructing hypotheses and models, and Big Data mining uncovers relations and patterns of the things we did not even know to look for (Dyche, 2012). Steadman (2013), in his "Big Data and the death of the theorist," concludes that the true objective of Big Data is to predict rather than understand the world; thus, *prediction trumps explanation.*

Data are not valuable in themselves, and analyses make them so; for instance, companies can predict the future better and make a profit by correct analyses of customer behavior and global financial trends. Hence, no one can ignore the

stories of saving billions thanks to data analyses (Marr, 2016a).[5] Then, it is not due to only Big Data that data are so much valuable; rather, data deluge and Big Data analytics enable us to take out the valuable outcomes: the more data, the more discoveries. Then, the more powerful Big Data analytics we have, the more value can be mined from the data deluge. In addition to all said, since the dynamics of the data have changed, data analytics must also change. Here, the field that includes traditional data analytics also develops and implements tools that can process Big Data is called *data science*.

1.1.2 Data Science and Unstructured Data

Data science is an interdisciplinary enterprise requiring expertise in statistics, modeling, machine learning, analytics, computer science, philosophy, and domain knowledge. Its main aim is to make sense of abundant data stored in various forms that cannot be handled with standard terms. The statistics about Big Data volume often denote the *velocity*: such volume of data is generated at high rates. Velocity is an issue when there is demand for real-time analytics. Thus, Big Data analytics, or data scientific tools, must provide solutions for grappling with data processing at enormous speed. Moreover, the feature of the *veracity* of Big Data, which is about trust and uncertainty regarding data, data source, and the results of data analysis, is critical. Suppose that more than two devices produce incorrect data in a system, and other connected devices perform their function according to those incorrect data, which eventually would be a disaster. Then, data scientists have to think in advance about how trust can be confirmed in a data deluge that expands constantly and rapidly; furthermore, they also have to meet the need for data analytics to manage and analyze uncertain data.

On the other hand, if the data is structured, none of the previously mentioned issues would require a novel approach to data. In other words, the chief problem that data science has to solve is that Big Data actually consists of unstructured

[5] One can refer to these stories from Marr (2016a).

data. The feature of *variety* of Big Data refers to the structural heterogeneity in a dataset (Gandomi & Haider, 2015). Structural heterogeneity means that data sets include the spectrum of fully structured data to unstructured data. Structured data refers to tabular data, e.g., CSV files; semi-structured data does not entirely accord with data standardization, e.g., XML documents; unstructured data lacks structural organization, e.g., e-mails, videos.[6]

Structural organization is an essential notion for data analytics analysis. Moreover, this is not a brand new problem in information systems. The experts have curated data to make them process-ready. Yet, we need Big Data technologies to come into play since traditional data management techniques can operate on structured or semi-structured data sets. In contrast, more than 90% of the data is unstructured. Converting unstructured data into structured one is costly and sometimes impossible with today's technology. That is even implausible since data comes in huge volumes in continuous mode. In Big Data, there is nothing like dispensable data since some results of the analyses in Big Data have shown that each data could have a role in leveraging power in "product promotions, placement, and staffing" (Gandomi & Haider, 2015, p. 138). Consequently, new data techniques must be developed, and new insights should emerge in order for Big Data to manage all kinds of data kinds, especially the unstructured ones, which is the main issue of data science.

1.1.3 Big Data and Industy, Science, and the Web

The very fundamental wheels upon which the world makes progress are science and industry. Developments in each field have resulted in new stages or revolutions to shape the world's destiny. Specific to our age, the improvements in hardware and software have changed how we produce: Big Data helps people

[6] Consider the following analogy. The books in a library are structured. Each book and each bookshelf is labeled, so by following the instructions, we can pick the book we want with ease. When we know the genre of the book and the bookshelves are labeled according to the genre, then we look for the book only on the related bookshelves. Then, this kind of library is semi-structured. When neither books nor bookshelves are labeled, this full-of-books-room is unstructured: to find the book we are looking for, we have to check each book until we find it.

in science and industry make discoveries. The Web, another field that dominates our lives, has gained a challenging role in Big Data. The following lines will define three essential fields of human life: industry, science, and the Web. What the data deluge has changed in these fields and what kind of revolution the impact of Big Data can occur will be discussed.

1.1.3.1 Industry

The term 'industry' points to economic activities that produce or provide "goods, services, or sources of income" (Encyclopædia Britannica, 2011). New technologies, the latest scientific discoveries, and/or unexpected social phenomena (such as the dramatic increase in population) spark industry revolutions. So, having always been one of the foci of human enthusiasm since the eighteenth century, Industry has gained another attention after introducing Industry 4.0 at Hanover Fair in 2011. That indicates three industrial revolutions occurred from the eighteenth century to 2011. Each industry revolution has reduced the need for a workforce in the factories and produced devices that surpass humans' skills and muscle power. For instance, robots have replaced workers on the assembly lines with the advent of *automation*. Robots are, in general, machines that are programmed in order to carry out a series of actions automatically.

The present-day technologies of the Internet, data analyses, Big Data, robotics, and alike introduce truly and fully automation from production to distribution processes. Moreover, the assemblage of these technologies is believed to transform automated factories into *autonomous factories*. That means increasing profits, decreasing costs, improving customer experience, maintaining newly introduced raw materials, optimizing lifetime value, and other market issues are held by the intensive assistance of the autonomous machines. This is Industry 4.0: "creating intelligent object networking and independent process management, with the interaction of the real and virtual worlds representing a crucial new aspect of the manufacturing and production process" (GTAI, n.d.). Such intelligence paves the way for autonomous decision-making in all aspects of marketing, namely in the design, production, operation, and service of products.

1.1.3.2 Science

Science, one of the highest human enterprises that guides the course of the world, is a complex system with methods to understand secrets of the phenomena. Its characteristics are systematic observation, experimentation, reasoning, construction of hypotheses and theories, and testing. Data collection can be held in a natural or laboratory setting or from simulations; reasoning methods can vary in different scientific works; the way of testing hypotheses can vary; yet the very aim of science, knowledge production, stays the same. Since the advent of computers and their involvement in scientific knowledge production, the machine has become an indispensable tool that has changed the way of doing science: without the software, some data could never be collected, and further, they would never be analyzed or from which conclusions would be drawn. Scientists have to rely on the results that the machine generates; therefore, the machine has become an indispensable component of scientific knowledge production.

Current improvement in the technology of experimentation and measurement yields a vast amount of scientific data. What has been revolutionary in science is that the machine is involved in data generation and analyzing processes; in other words, the machine has become indispensable in scientific knowledge production. Furthermore, the impact of Big Data and data analytics in science, namely the upcoming scientific revolution is when the machine behaves like a 'scientist:' it can systematically observe, conduct experiments, do the reasoning upon its findings, construct hypotheses, and test hypotheses. In other words, the machine becomes autonomous.

1.1.3.3 Web

Tim Berners-Lee dreamt that "all the information stored on computers everywhere were linked," and every computer could be programmed to provide a space in which everything could be linked so that "all the bits of information in every computer at CERN, and on the planet, would be available to" everyone (Berners-Lee & Fischetti, 2001, p. 4). In 1989, he realized his dream of creating

a single, global information space: the World Wide Web, or the Web. The Web is an indispensable component of our daily lives since then.

At the beginning of the Web, the 'space' was small, and a few people, who were professionals, created Web content. In those days, the users who knew the webpage address could reach and only read the pages. However, the static nature of the webpages changed when the 'space' turned into a place where people-to-people communication and dynamic data sharing could occur. However, a dramatic increase in the number of webpages and usage required the 'space' to become searchable. Besides, the Semantic Web technologies were introduced to organize the 'space' so that the space would consist of meaningfully related contents. Big Data has greatly impacted the 'space' as well. As of May 26, 2022, there are at least 4.87 billion indexed webpages.[7] The Semantic Web technologies are too old to deal with the new space. Thus, the Web data should be organized to warrant automation, integration, inferences, and integration with data analytics that promise prediction, personalized web experience, and alike. In other words, Big Data technologies aim to bring the pages that the users are interested in or may be interested in.

1.1.4 A Bestow of Big Data and its Technologies: Autonomy

The effects of Big Data are so transformative in Industry, Science, and the Web (ISW). The future of Industry is smart factories, where the machine automatizes each stage from good productions to distributions. For this to happen, there needs to be interoperability in human-machine communication and decentralized intelligence in information-intensive manufacturing. It must be decentralized since making each device talk in the same language is impossible. It is crucial that the devices can understand each other and also understand the data in Big Data.

[7] See WorldWideWebSize.com.

The machine will be both knowledge sources and agents determining the produced knowledge's status in science. The future of Science demands that the machine contributes to scientific knowledge production by making recommendations to scientists, answering queries, and explaining the patterns as an autonomous workfellow. Otherwise, there would be no contribution of Big Data to science.

The future of the Web, on the other hand, presupposes the machine to become content managers and producers. In the former, the machine will answer the natural language queries in any subject; in the latter, the machine will generate content for a task from the Web's voluminous information. However, the task prerequisite is the machine's ability to classify the webpages for a given search/query. To wit, the Web will be a giant knowledgebase upon which the machine will classify the webpages and then make inferences and generate content for a given topic.

Accordingly, and in conclusion, the most significant impact of Big Data on ISW is that it transforms the machine into an *actual agent*: the more there are data, the more influential the machine is. Therefore, data science aims to create algorithms that can decide, make inferences, cluster the contents, predict, recommend, and exhibit alike higher cognitive faculties. Namely, they should make the machine *understand Big Data.*

1.2 The Most Vital Issue of All: Processing Semantics

In the previous section, we found that machines need to be autonomous in order for Big Data to benefit ISW. We have mentioned that the autonomous structure also depends on developing a system that understands the machine. This section detects the problematic aspects of the last one.

We mentioned what role the machine should gain at this point: being autonomous, which is associated with the machine-understandability of Big Data. Now let us examine the processing of Big Data in these fields and discuss whether Big Data leads to a revolution.

1.2.1 Processing Big Data in ISW

A shift from factories with automation to smart factories is said to be possible with Big Data and its technologies. It is necessary to mention the Internet of Things (IoT) here because the communication of electronic devices in the factory and even outside the factory is provided by IoT technologies. So, these devices generate and interpret a great deal of data. However, smart factories cannot be limited to robots' communication in the factory. Since they are called "smart," they also interpret data from outside the factory, for example, data on demand for what the factories produce. All these data are in the Big Data status regarding volume and diversity. So, what the machines in smart factories do is process Big Data and act accordingly.

We can utter similar things about the position of Big Data in producing scientific knowledge. A scientist no longer looks at the sky with a telescope but at the data on the computer screens sent to computers by telescopes in space. Moreover, this scientist uses software to analyze the data coming from the telescope. Besides, the machine is not only responsible for collecting phenomenal data but also for producing them with simulations. In addition to the data, machines collect and produce, and scientists write reports and papers on their subjects. Thus, a massive collection of scientific data is stored in the machine. Processing that much data, as a whole, is not something a human scientist can do. In that case, the machine needs to be able to process these data to benefit from Big Data. For this reason, the machine should be expected to collaborate with scientists, almost like a colleague: reviewing the literature, analyzing the data, forming a hypothesis, and sharing the findings.

Undoubtedly, the Web is the field where the impact of Big Data can be grasped most easily. While a certain number of webpages were coming for research decades ago, now the search results listed in millions of pages come as images, videos, gifs, audio, and text. However, in order for the Web to work more effectively, it needs to go beyond just the classification of images, videos, or texts: it should classify all the sources on the Web related to the searched term

according to their content. Thus, Big Data expands the Web; the machine for the Web controls Big Data.

The following will discuss the limitations and biases of Big Data and question whether smart factories, lab colleagues, and content controllers can grow up in Big Data.

1.2.1.1 Biases of Big Data

"Big Data speaks for itself free of theory" is the motto for some scientists and data scientists. The underlying understanding of this claim is that there is a new epistemology of data, which is more comprehensive, objective, and productive.

Big Data is more comprehensive; the more data, the more discoveries. To this end, and thanks to Big Data, there are data from any granularity, so nearly all aspects of all domains can be captured. For instance, Rieder and Simon (2017) claim that the complete picture of a domain is available, but before Big Data, data gathering in scientific inquiries was limited by the hypothesis or models. Moreover, such unrestricted sampling favors that the same phenomena can be analyzed from various aspects, and the researchers benefit from the comprehensiveness provided by the data deluge. New data analytics, thus, shed light on more results, which were previously limited by specific data sets and modelings (Mayer-Schoenberger & Cukier, 2013; Steadman, 2013).

Further, the data is believed to be raw, objective, and neutral.[8] Data generated by machines, by us, or both are not within the scope of narrow observations; rather, they are gathered from various sources without a limitation. Thus, Big Data is free of human bias and framing. Hence, data sets are inherently meaningful (Kitchin, 2014). Indeed, not only is the data free of human bias but the new analytics as well. Rather than being constructed for a particular domain, "agnostic data analytics" are tools that can be applied to various data sets from various fields (ibid.).

[8] See Gitelman (2013) for other beliefs. Cf. Rieder and Simon (2017).

Hence, the same results that raw data meet agnostic algorithms are (i) data speaking for themselves free of theory and (ii) patterns and relations being meaningful and accurate. Since neutrality is sustained in data gathering and analysis, this new empiricist way of knowledge production promises the triumph of correlations. Thanks to a huge volume of data, "correlation is enough," and "data without hypotheses about what it might show" are analyzable (Anderson, 2008). In this respect, there would be no further need for causal explanations. It is because we are not to understand the world as itself, but rather to predict the phenomena in the world, as the correlations are proficient at providing "results:" we moved away from the old-fashion search for causality. More precisely, in scientific practice, correlations are taken as something to be interpreted, yet Big Data surpasses these concerns: correlations are more informative and evidential (Mayer-Schoenberger & Cukier, 2013). Even more elucidator since Big Data reveals correlations, even the ones that we did not think of (Dyche, 2012) and new analytics "find patterns where science cannot" (Anderson, 2008). Prensky (2009) reads Anderson's-like claims as "scientists no longer have to make educated guesses, construct hypotheses and models, and test them with data-based experiments and examples," and "without further experimentation," they can reach patterns that should be taken as "scientific conclusions" from the data deluge.

To summarise, Big Data analytics provide a new mode of knowledge production, whose aim is making sense of the world by obtaining *insights* and *knowledge* embedded in data; rather than the classical mode of scientific inquiry- hypotheses, model, and test by analyzing the relevant data. Hence, (1) the machine knows the patterns it draws and the predictions it makes. (2) Machine knowledge, namely pattern findings and/or prediction power, is limited by data and the algorithms we generate. Due to data deluge and powerful algorithms, on the other hand, a machine can know *anything*.

1.2.1.2 Limitations of Big Data

There is an indisputable truth here: there exists data deluge in all domains, and continuous advances in machine power enable the use of those data that was hitherto impossible. To wit, the impact of Big Data and its technologies are crystal-clear. However, the previous motto should be substituted with "The data is mute:" data cannot be comprehensive, objective, and productive by itself.

First, more data does not mean more discoveries or better inquiries (Symons & Alvarado, 2016). Leonelli (2014) states that there is no revolutionary role of Big Data in some fields, e.g., in biology, because different epistemic communities use different methods, materials, and background information for knowledge production. This requires format standardization in various fields within biology, which is an expensive task. Moreover, such standardization also requires data curation, which is both expensive and labor-intensive.

In light of the last paragraph, we can say that structuring Big Data is hardly possible. We purport two reasons. Firstly, as Leonelli (2014) states, it is a tall order. Any structured data representation, like databases, taxonomies, or ontologies, suffers from interoperability problems. These representation products are designed for specific problems, for specific domains, and/or from specific perspectives. As these are software products –not just an intellectual exercise, they are required to meet the needs of the companies/tasks/customers. Thus, integrating these representations means providing a constellation, which is hardly possible. For instance, an extra ontology that glues others is required to provide ontology interoperability. That ontology can be a bridge ontology that is designed explicitly for the ontologies that are supposed to be unified, or it can be a reference ontology that rules over the domain. However, this approach is not economical. Secondly, data structuring tools work on specific domains, and their products, such as databases, ontologies, ER models, and cybernetic systems, are static in their nature. On the other hand, Big Data is dynamic in its nature. For instance, the introduction of a new device in IoT is inevitable. And accordingly, data are structured in these systems as centralized. That means labeling the entities and relations of a context is purposeful. However, Big Data requires a

decentralized structuring since the meanings of entities and relations can vary inter contexts. Even if all people in the world come together, they cannot make Big Data structured using existing technologies.

At this point, a caveat is needed. A big mistake is that 'machine-readability' and 'machine-understandability' are used interchangeably in the literature. Such a mistake is the opinion that machine understanding is possible by linking the tagged terms. For instance, Tim Berners-Lee (2001, p. 185) defines machine-understandability in the sense that building understanding enables to link 'very many meanings.' He continues that concepts are linked together by "frequent contributions from independent sources" (ibid, p. 187). On the other hand, linking concepts together is not enough to create a meaningful Web. Each context specifies the meanings of entities, and relations are specified differently in each context, which means an entity can be in different contexts with different meanings. The collections of different meanings cannot constitute the entity.[9] Or consider an example from the industry. A device receiving signals from various devices is to *understand* what each signal means and then performs accordingly. At this rate, signals have epistemic value, and the machine has to interpret them and perform accordingly.

Secondly, the ideas that huge amounts of data can wipe out the need for a priori; accessing large data sets secures the automatic elimination of human bias and framing and error; the algorithms are so powerful to drive insights or knowledge from the data deluge by data analytics are hardly provable. Marcus (2018, p. 5) criticizes these ideas and mimics them as "in a world with infinite data, and infinite computational resources, there might be little need for any other technique." However, Marcus analyzes the alleged dexterity of deep learning and emphatically states that data are never infinite, and algorithms are not that powerful. Each deep learning algorithm is suited to a specific task, and each can learn bias from the data they use. Moreover, algorithms may maintain

[9] Here, 'entity' refers to objects/things, not to relations or event. On the other hand, in this work, the term 'entity' refers to anything in the world; a quark, the beauty of a line of code, an irrational number, swinging, a yellowish star, Uzun İhsan Efendi, to solve the issue of gender discrimination in developed countries, and so on.

the prejudices of their designers (Rieder & Simon, 2017; Cf. CIHR, 2015, cited in Rieder & Simon, 2017). "[...] [S]ystems are designed to capture certain kinds of data and the analytics and algorithms used are based on scientific reasoning and have been refined through scientific testing" (Kitchin, 2014, p. 5), such requires a theory behind it. Then, the algorithms are not agnostic at all.

Thirdly, it is incorrect to accept "the unrestricted correlation," where correlations are regarded as truths born from data (Symons & Alvarado, 2016). Correlations cannot be taken as facts, and neither data analysis nor its interpretation is intrinsically unbiased; rather, results driven by data are in some conceptual framework. So, making sense of data is always framed. Leonelli (2014, p. 4) states, "[...] no matter how large, [data collections] are diverse enough to counter bias in their sources. If all data sources share more or less the same biases [...], there is also the chance that bias will be amplified, rather than reduced, through such Big Data." In light of this, one can utter that bias is indispensable in Big Data analytics. To wit, the mechanical objectivity of Big Data is questionable in both data deluge and its use.

Furthermore, data being raw, objective, and neutral are controversial. That is counterintuitive because the machine just records data without an intervention. Gitelman (2013), however, illustrates that objectivity, truth, and trust are considered as situated in the history of science and technology. Data being raw is also criticized by Bowker with the following words: Raw data is both an oxymoron and a bad idea; to the contrary, data should be cooked with care (2005, as quoted in boyd & Crawford, 2012). Hence, Big Data is not necessarily raw and objective. All of that, the data provides a specified view of the world using particular tools, for collection of data starts with the identification of the conceptual framework of the inquiry. Data cannot be gathered together as they are in themselves; hence, the data deluge is neither neutral nor free of bias or framing.

Hitherto we have seen the alleged features of Big Data. Now let us dive into whether Big Data tools tame and cope with unstructured data and meet the needs of ISW.

Today, machines not only record excessive observations and generate inductive evidence through simulations. Nevertheless, the machines do methodologically gather more data (Frické, 2015), which cannot be taken as scientific discovery. Patterns are necessary but not sufficient for scientific development; strong correlations are not the harbinger of causal links between phenomena. Marcus (2018) urges us that deep learning – the deified model for every task, a statistical technique that uses neural networks with multiple layers to classify patterns- algorithms learn from complex correlations between input and output values, yet cannot represent any inherent causation between them. In Industry and Science, we need causal explanations; this is how science works and how we can trust the machine's decisions.[10] Again, patterns are not inherently meaningful,"[a]ny analysis would be better with priors, Bayesian prior probabilities, or similar, for any hypotheses" (Frické, 2015). Pearl and Mackenzie (2018) argue that data by itself is "dumb," and it can only tell correlations and associations but can never tell us why; this is the reason why we should dissuade ourselves from the charm of the so-called superhuman discoveries of data analytics. That is, the machine cannot cluster the Web contents according to searched terms when it is trained with user habits. Consequently, no matter how large datasets we have, there is nothing like the primacy of correlations over causality; Big Data cannot wipe out this fact.

Marcus (2018) further challenges those who support those statistical and probabilistic models are satisfactory in Big Data. Given its due in speech recognition, image recognition, and other classification problems, Marcus challenges deep learning from several aspects. Firstly, deep learning is not able to learn abstractions through explicit, verbal definition; moreover, it needs to crunch millions or billions of data to provide satisfying works (p. 7). Furthermore, the patterns that deep learning extracted are too shallow, and deep learning cannot deal with the hierarchical structure of the language (p. 9). Besides, such models cannot relate the background knowledge or open-ended inferences (pp. 10–12). Above all, deep learning models are black boxes, and their correlation-based nature

[10] Pigliucci (2009) criticizes Anderson for being ignorant of science and the scientific method.

makes them unreliable tools at all (p. 11; p. 13). As of 2022, no deep learning model copes with these aspects.

Additionally, one may say that deep learning is about classification and may even criticize us for speaking about deduction. This relevant claim can be defended by the fact that deep learning cannot classify abstract concepts (Cf. Marcus, 2018, p. 7), and the non-hierarchical structure does not allow property- and relation-transfers. In order for the machine to employ deduction, algorithms should construct classifications, which in turn enables the subsumption relation to function so that the child class can subsume attributes of the parent class. That is to say, deep learning algorithms cannot classify but cluster concepts. Hence, the fabulous deep learning models can neither represent nor generalize the rich structure of the world reliably. At last, Big Data analytics cannot promise a revolution in ISW.

1.2.2 Structure of Big Data

Present-day machines are far from being active agents. They are just expert systems constructed with structured data of a specific domain for a particular task and/or statistical models using structured and/or unstructured data. Decisions/recommendations of the machine are valid only in the domain represented or trained for the machine. That is, the produced results can be inapplicable to another domain, and/or the representations can be insufficient for the other. However, the machine is supposed to make decisions and recommendations from Big Data, which is not limited to a specific domain. Machines in the smart factories have to derive decisions from various environments; the machine has to cluster contents from various contexts on the Web; the machine has to communicate automatically between various scientific works of different aims.

The state-of-the-art methods are no longer feasible for processing knowledge in the depts of Big Data; namely, they cannot go beyond knowledge retrieval or extraction from shallow structured data. Some of the biggest challenges in front of knowledge processing are that most data are unstructured; semantic properties are annotated manually; knowledge is represented in a centralized fashion,

and formal tools are so limited for operating on various reasoning methods. As a consequence, Big Data and its technologies do not have a transformative effect on ISW as they do not provide a framework on which the machine can be autonomous. So, if the very nature of Big Data is unstructured, then how come can an automated system make decisions and recommendations and provide explanations with unstructured data? In other words, what should change so that Big Data can be utilized and the machine be autonomous.

1.3 Building a Machine that Understands

For Big Data to revolutionize ISW, machines need to be autonomous; for machines to be autonomous, they need to understand Big Data and make inferences from it. This research sought to foundations of a machine-understandable system.

In the previous section, the difference between machine-readability and machine-understandability was slightly mentioned. This issue is vital because we need to clearly state what we mean by a machine-understandable system to identify the foundations of establishing such a system. Machine-readability is the machine's ability to process data; in other words, machine-readable data is structured data. For instance, suppose that we need to inform the machine about faculty members' personal and institutional information and ask the machine who lives outside the city center and has morning classes. Writing all the information as a text file is not machine-readable since the machine sees just a bunch of symbols, which the machine cannot employ operations; because it cannot detect data values and types. However, creating a spreadsheet can solve the problem. Creating columns specifies what information is collected, and creating rows tells the machine what values the columns get for each person in the faculty. From this structure, the machine can employ an operation to find out person(s) who has/have morning classes and live(s) outside the city center. Or the same scenario can be structured with ontologies, in which each entity will be classified, and relations between classes and individuals will be specified. This kind of representation has semantic components, such as transitions between classes and

relations can be reflected here. Thus, machine-readable data is the data with semantic values that are attached to data.

Machine-understandability is more than machine-readability. Here is why. Machine-readable data has limitations in decision-making since decision-making includes non-contextual and contextual-but-implicit relations. That means, when a context is structured, the semantic values are fixed to entities so that the machine operates on these static semantic values of data. A machine understands when it can detect/choose/find other semantic properties of entities. For instance, suppose that the context about faculty members is structured through an ontology. A faculty member is known to have morning classes, but their address information is missing. However, somehow in the ontology, they are linked with Bus 589 –which goes directly outside the city. If there is information about bus routes and numbers, the machine can infer that that faculty member comes to the university in the morning from outside the city. When all this information is structured, no worries; however, these are toy examples; we aim to structure Big Data. Thus, machine-understandable data should be a data format that includes all the potential semantic values an entity could take in any context.

Data are processed regarding their values and types. On the other hand, semantic data, or structured data, are analyzed through their values, types, and attributes/semantic properties. In other words, semantic properties are the *tagged* aspects of the data. Tagging *all* the aspects of an entity is undoubtedly a daunting task; nevertheless, we have to change our perspective of attributing semantic properties to entities. In that case, the machine should attribute semantic values to entities according to context. At this time, the machine is said to process entities dynamically.

Before moving on to the analogies we have for entity representation and processing, it is necessary to pause here for a caveat. All data structures, for instance, ontologies, are machine-readable software created in structures humans can understand. Machine-understandable does not have to be human-understandable. In other words, humans do not need to understand the structure of Big Data. Zambak (2014, p. 69) defends "the idea that machine intelligence does not nec-

essarily process information in the way humans do." In the line of this idea, we claim that humans and machines do not necessarily share the same categorical foundations for depicting the world; thus, we must carry out our work from a machine perspective.

1.3.1 Representing Entities

We believe the real trick to creating a machine-understanding system is increasing the representation power. The higher the number of semantic properties that the machine can process, the stronger the inference ability. For example, imagine that there were only natural numbers with which all the scientific works were carried out. Scientists were only accepting computations and results of natural numbers. There were days when scientists could not, and today need not operate with numbers other than natural numbers. However, if we only had natural numbers, science would not have reached its glorious position in human lives. However, when other kinds of numbers, such as real numbers or complex numbers, were introduced, representation power increased, and science has produced much more profound information about phenomena. Thanks to science, advances have been made in the history of humanity. So, if we expect the machine to take part in processing the meaning in Big Data –that is, in making decisions, recommendations and analyses– it must process the semantic values of the data, and thus the entities must be, in principle, represented in terms of all their semantic properties. We suggest that semantics must be involved at the data beginning of the data processing. In light of this, we purport that entity representations must include all possible semantic properties so that the machine can choose among them.

Consider the analogy that just as living things are the sum of certain features, these features are represented in specific compositions in the DNA sequence, so entities are represented as the composition of semantic properties, and the machine can select the semantic properties from the composition according to contexts. So, in this dissertation, it will be investigated how this representation should be.

1.3.2 Processing Context

We claim that the context determines the meaning of the entities. This has twofold results. The first one is that the machine is supposed to process contexts, not entities. The rationale behind this is that entities gain their meaning within the interaction with other entities in a context. On the other hand, this does not mean that keeping the same entities always yields to the same context since a new context means the re-establishment of all forms of interactions. Think of a doctor and a nurse in an examination room of a hospial. The roles and behaviors of these people are clear. However, these roles/behaviors are about to change when the nurse is a patient, and the doctor is taking care of him. The people and the place is constant, but the form of interactions has changed. Thus, the machine must be able to detect such interactions. That is, the machine determined the ontological statuses of the entities in the context of their interactions. Let us unfold this assertion.

The best way to understand an entity is to understand its relations to other entities. Entities by themselves cannot exhibit their roles in a context. We can use Figure 1.1 to illustrate what we mean.

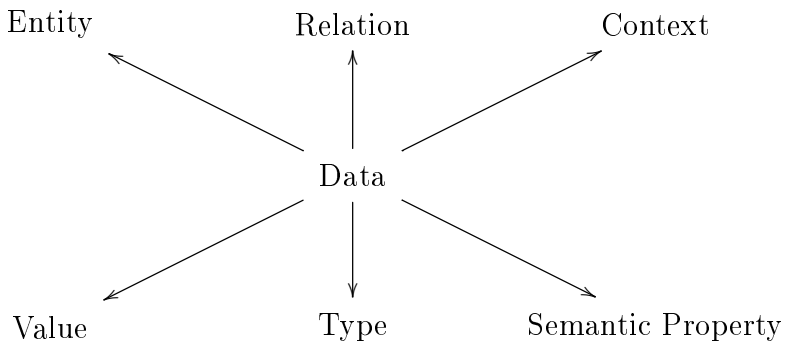


Figure 1.1: A weird analogy mirrored by Data

It is the data value that is processed, and how that value is processed is decided based on the type of data. For instance, imagine data whose value is '3.' If the data type is integer, this data value can be added to another number. But if its type is character, then the addition operation cannot be applied but rather the concatenation operation. When the data is machine-readable, we expect the machine also to process its semantic properties. In other words, the values

are operated according to the type and semantic properties of the data. A kind of relevancy appears between value and entity, type and relation, and semantic property and context in Figure 1.1. Because in a structured system, which rules/axioms are applied to an entity are determined by its relations, and its meaning comes from within the context. When we use this analogy to a machine-understandable system, the contextual relationship types decide which semantic properties of an entity will be processed. However, since we claim that entities are represented as the composition of the semantic properties, in this case, it will be necessary to decide which semantic properties of an entity will be processed through the semantic properties of other entities. This brings us to the second result that entities must be represented in terms of interactions.

Determining and processing relations is the key aspect of this dissertation. In other words, when the machine can detect and process relations, it can handle the contexts. A paradigm shift is indispensable at this point: this work needs to change the orthodox entity prioritization over context. This means that the ontological status of the entities in the context is automatically determined by the way they are associated, and inference can be made through this structure, where all possible semantic properties of the entities can be represented. This, above all, means that there must be a new kind of ontology that is particular to machine-understandability.

1.3.3 A Machine Ontology

On the way to the research aim, finding machine-understandability foundations, we encounter a new research objective: creating a machine ontology. It should be such an ontology that it would not be *data without ontology*, as in the early days of informatics. Or, it would not consume human labor for labeling to transform the entire data sources into a semantic structure; that is, it would also demolish the *data with ontology* approach. Let this ontology be *data within ontology*. Let it introduce the world in ontological relations, create ontologies according to the context, and process these ontologies. Thus, let it transform phenomena into

data, incapsulate background information, and finally and above all, make the machine understandable.

Let us mention some points to consider in creating this ontology. First of all, this ontology should prioritize relations, not entities. That is, it must represent the phenomena itself in the machine through relations. Second, it should define entities as the composition of their semantic properties. Then, semantic properties and their compositions should also be represented ontologically through relations. Next, since this ontology is relation based, there must be an axiomatic structure for how compositions are created. An example is isomerism from chemistry. Isomerism is the phenomenon in which more than one compounds have identical molecular formulae but distinct arrangements of atoms in space. Isomers do not necessarily share similar chemical or physical properties. Thus, entities cannot be represented as a bunch of semantic properties since there can be entities having the same semantic properties, but they can be distinct. Of course, it must be reminded that all these claims are from a machine perspective, and the machine cannot have to have a complete picture of the entities. For this reason, this ontology should also bring together the semantic properties in a certain rule/structure and define the entities. Consider the following analogy. All humans have a DNA sequence; what makes each of them the individual is the order of nucleotides in DNA. Moreover, humans and bonobos share approximately 98.7% of their DNA (Max-Planck-Gesellschaft, 2012). Just a 1.3% difference in the DNA results in cognitively distinct creatures. This example is provided due that even if this 1.3% difference did not exist, the machine ontology we aim to establish should be able to identify them as different entities by looking at how humans and bonobos relate to other beings.

The rest of the research objectives are a formalization of the mentioned ontology and how the machine can make inferences through the ontology. Before our exploration, let us note our first impressions of these objectives.

Whatever formalization tool will be used, some rules and axioms need to be added to the formal system, as these tolls are not designed for the ontology we intend to establish. Only then can an accurate inference system be obtained.

The most crucial point to consider in the formalization of a machine ontology is that a set-theoretic approach will not be accepted. Because in a set theory, entities are prioritized. Then this work will find formal systems that have abstractions through relations. Moreover, a formal system that can formalize a machine ontology must have higher levels of abstraction since we aim at representing phenomena in data. Such a tall order has such high demand.

Creating autonomous machines is the prerequisite that Big Data benefits Industry, Science, and Web. This work aims to understand the foundations for machine-understandability, which is believed to pave the way for machine autonomy. In this respect, this work aims to create a machine ontology, which represents entities in terms of their semantic properties in a relation-based fashion. This objective is twofold: an appropriate representational framework and formalization tool will be found.

1.4 Some Facts About the Dissertation

This work will benefit so many research and application fields. First and foremost, this work has developed an alternative perspective to formal ontologies and applied ontologies. The studies on these fields prioritize objects over relations and accordingly often use set-theoretical formalization. Hopefully, this work will encourage those who work on formal and applied ontologies to escape dependency on and boundaries from substance ontologies and set-theoretical frameworks. Secondly, it has been pointed out that it is essential to establish an ontology of machine-understandable, not human-understandable. Thus, a new ontological perspective for general artificial intelligence is presented. This study, an example of applied category theory, will also bring a category theoretical perspective to the ontology research and development of ontology-based systems. In addition to these, it is also an example of the contribution of the applications of philosophical teachings to information systems. In other words, this work draws attention to the contribution of philosophical analyses to developing technologies in machine/general artificial intelligence. It highlights the contribution of philosophical analysis to the development of technologies. There

is still some knowledge to be taken from philosophy; however, it should be noted that this is not one-sided. The needs for machine intelligence also enable the enrichment of philosophical attitudes. So, this work provides an example of a twofold contribution that will be used for further research.

In addition to the contribution to academia and research, the findings of this work will elevate all the data-driven fields. That is to say, and even more precisely, this work will bring about a revolution in data science. This work will benefit, for sure, Industry, Science, and the Web, and also other data analytics fields, including content recognition, text generation, question-answering systems, content clustering, to name some. The real impact of Big Data will emerge with this study.

Said all, however, this dissertation has some limitations. First of all, this work is a proposal rather than a full sketch of the architecture of Ontology 4.0. Consequently, implementation of it can have unpredictable impediments. The reason for this is the lack of experience in applied category theory. To make a possible limitation more concrete, tropes and their compositions may be represented as toposes with some extra features that are particular to the urtrope theory, or another constructor can be found to represent urtropes that are specific to the urtrope theory, or there can be other constructions that can represent tropes as typed categories.

1.4.0.1 The Structural Outline

Chapter 1 has introduced the content of this dissertation.[11] It provided the background of the work by defining data, Big Data, and data science. It introduced the three most important fields that shape human history: industry,

[11] The structure of this work was planned to construct a picture of industry, science, the Web, data, and ontology about their gradual developments, changes, or kinds since the work is in the intersection of research areas of philosophy, mathematics, computer science, data science, and information systems. However, the construction was changed due to the need to lead in the thesis immediately. So, we recommend reading the first two appendices that have an introductory role. Moreover, the ones who are not familiar with applied ontologies are strongly encouraged to read section Ontology before Chapter 2.

science, and the Web. Then, it discussed the failure of the expected impact of Big Data in these fields and explained that a revolution through Big Data and related technologies could be achieved by finding a solution structuring data in the wild.

Chapter 2 will examine the effects of Big Data on the transformation of ISW through the difference between the current situations and the situations in which ISW pretend to be. That the machine being autonomous will create a real revolution will be argued. Moreover, it will be shown that it will be possible for the machine to be autonomous with the modifications to be made in the data structure. In this respect, it will also examine the existing data structures in classification and search for the type of data in which the meaning will be processed.

In Chapter 3, how the machine-understandable data should be represented will be examined within philosophy. The findings will investigate how the created data representations will be implemented on the machine and how they will be processed. A philosophical foundation for a machine ontology, viz., the urtrope theory, will be introduced in this chapter.

Chapter 4 will introduce category theory, a tool to formalize the urtrope theory, and then give general information about it. This chapter will also assert how to represent ontological categories in the urtrope theory in the category theoretical parlance. Lastly, Chapter 5 will summarize the essential components of this dissertation. Based on these components and the analyses on machine-understandability thereof, the features of Ontology 4.0, the gist of the dissertation, will be particularized to rectify the alleged revolution of Big Data in ISW.

CHAPTER 2

AUTONOMY AND DATA

Chapter Introduction briefly challenged the impact of Big Data on Industry, Science, and Web. This chapter addresses these challenges by finding out where they are actually about to reach under the spell of the transformative effect of Big Data. The difference between their current situation and the situation they pretend to be will be examined. Moreover, what features the machine should have so that ISW can finally reach the state they want to achieve will be investigated. Then, the focus will be shifted toward data whose nature will be examined.

2.1 Previous Versions of Industry, Science, and Web

This work attempts to divide Industry, Science, and Web into phases from the machine and/or data involvement, generation, and/or usage perspectives. These features cannot be taken rigid; rather, they help illustrate the current and future positions of these fields from the aims and boundaries of this work.[1] Indeed, there is a chronological aspect of phases, yet a new phase may not annul the previous one(s). Said that, Appendix From 1.0 to 3.0: Industry, Science, Web gives a complete picture of previous versions of Industry, Science, and Web. This section summarizes it as to prelude to the fourth versions of ISW.

[1] At this juncture, we declare that we are not directly committing to philosophy or history of science, industry, or the Web; rather, we provide a picture that includes all the essential notions of the purpose of this work.

In general terms, industry is about providing services or producing goods for sale. In this respect, the beginning of industry dates back to the story of civilized humankind. On the other hand, the machine-involvement in industry and the punched cards, as encoded data, in the earliest textile machines revolutionized the industry. Known as the *Industry Revolution*, this is Industry 1.0. The emergence of mechanization in Industry 1.0 led to factory systems where electrical power was utilized, and new machines were invented. So, introducing the assembly line that called forth mass production initiated Industry 2.0. The third period is called the *Digital* or *Computer Revolution*, which started with the advent of computers. The new roles and the characteristics of data and the machine led to automation in factories.

Science is one of the highest human enterprises that shape the world. Dividing the history of science into some phases or periods can vary from scientific and/or philosophical points of view or different approaches. Here, we are dealing with the history of science from machine-involvement in the scientific knowledge production approach. In this respect, in Science 1.0, the then scientists were passive observers and conducted their experiments *in vivo*, and they generated scientific results by reasoning alone. The scientists in Science 2.0 conducted their experiments in both *vivo* and *vitro* since this phase began with the inventions of measurement tools and the experimental instruments that enabled unobservable becoming visible and also increased the volume of collected data. The scientists in Science 3.0, on the other hand, depend on the machine to carry out their experiments. The advent of computers caused them to measure the parameters in the experiments, process data, and contribute to data collection. Moreover, computer simulations have enlarged the scope of scientific experiments to *in silico*.

The story of the Web starts with a dream of Tim Berners-Lee. The dream was to create a 'space' where the scientists in CERN could share information, keep track of projects, reach technical details of ex-projects, or find recorded information stored in personal computers. Historically, Web 1.0 is the first kind of Web where all webpages are static; then, the users can only reach the webpage and read the contents created by only the professional content generators. As the idea of the

Web necessitates social interaction, Web 2.0 emerged so that people-to-people communication and dynamic data sharing have occurred. The web content has started to be also generated by the users. However, the considerable increase in the volume of web content entailed a work of organization of the content on the Web. Web 3.0 is the phase of the Web where the machine can read the content to analyze the data on the Web.

Industry, Science, and Web are the most important fields, so a paradigm change in them has the power to change the world. Should Big Data cause a paradigm shift in these, they need to reach their upper version. Now, let us examine the fourth phase of each field in turn.

2.2 Industry 4.0

The industrial revolutions have freed us from muscle and skill-intensive labor. In Industry 1.0, using steam power to mechanize the production process, we were freed from works that demanded muscle power; in Industry 2.0, making use of electricity in factories, we have been freed from the knowledge and skill-intensive labors; lastly, in Industry 3.0, robot technology has freed us from our physical abilities.

The present-day technologies of the Internet, data analyses, Big Data, robotics, and alike are freeing us by introducing truly and fully automation in the production process. In some sense, such truly and fully automation can be taken as *autonomy*, where the autonomous industry is information-intensive manufacturing. Inevitably, increasing profits, decreasing costs, improving customer experience, maintaining newly introduced raw materials, optimizing lifetime value, and other market issues are held by the intensive assistance of autonomous machines. Now, we will investigate whether or not these developments have caused a revolution in the industry. In other words, we will examine the kept-harping-on-about-Industry 4.0.

Hermann, Pentek, and Otto (2016, p. 3928) point out that there has been no clear definition for Industry 4.0 and that we still lack its detailed account as of

2021. Nevertheless, we can still argue that there are two main camps defining Industry 4.0 distinctly. The first camp does not take Industry 4.0 as a revolution but rather as a continuum of Industry 3.0 with developments in automation, productivity optimization, and data exchange in manufacturing. According to this camp, digitalization in manufacturing follows an entirely new way –that is why it is not Industry 3.0 no more–, which has been realized by machine learning algorithms, data analytics, and Big Data. Eventually, the ultimate product of Industry 4.0 is smart factories supported by Industry 4.0 components, such as the Internet of Things, cloud computing, and cyber-physical systems. The second camp, on the other hand, considers Industry 4.0 as a future revolution with autonomous factories and decentralized intelligence in manufacturing. Industry 3.0 to Industry 4.0 is a technological shift from embedded systems –centralized production– to a web of cyber-physical systems –decentralized production that handles all the processes from obtaining raw materials to meeting customer satisfaction. Regarding decentralized intelligence, Industry 4.0 is "creating intelligent object networking and independent process management, with the interaction of the real and virtual worlds representing a crucial new aspect of the manufacturing and production process" (GTAI, n.d.). Such intelligence paves the way for autonomous decision-making in all aspects of marketing, namely in the design, production, operation, and service of products. Consequently, according to this camp, smart factories are a part of Industry 4.0, and Industry 4.0 can be named Smart Industry.

In either case, Industry 4.0 is defined *a priori*, unlike other industry revolutions: we have an agenda to bring about a revolution. In other words, we are trying to reach the recommendations of implementations or standards of so-called Industry 4.0. Although there are many components of Industry 4.0,[2] within the boundaries of this work, we only focus on the two principal components of it: Cyber-Physical Systems (CPS) and the Internet of Things (IoT) (Cf. Kagermann, Helbig, Hellinger, & Wahlster, 2013, p. 5). So, we have to ponder on

[2] For instance, Internet of Services, Internet of People, smart products, M2M (machine-to-machine), cloud technologies, integration of information technologies, and operational technologies. Note that these components can be used interchangeably or refer to each other with different names.

whether or not we have the technology of *information driven cyber-physical environment.*

2.2.1 Cyber-Physical Systems

Cyber-Physical Systems (CPS) are the "integrations of computation, networking, and physical processes" (The Ptolemy Projects, n.d.). The twins, the physical and computational systems, which embody CPS, work reciprocally. The physical systems refer to the devices/machines, and the cyber systems refer to the computational models of these physical machines. The physical machines are monitored and controlled by cyber machines, whereas the computations in the cyber systems are affected by feedback loops of physical processes. In a particular CPS, the twins communicate in a closed system; on the other hand, a combination of those twins creates a network and enables manufacturing systems that communicate with each other. In that case, in the network of such devices, cyber machines can make some decisions on the manufacturing processes; and, further, make the physical ones execute in a dynamic schedule.

2.2.2 Internet of Things

The Internet is the backbone of Industry 4.0. The Internet of Things (IoT), the second fundamental component of Industry 4.0, consists of electronic devices connected to the Internet. These devices are embedded or attached technologies to perceive, collect and then execute the data for particular operations. In other words, an IoT device collects data to serve a particular purpose in a network of devices. An example can be a health monitoring system, say a watch. Let it be connected to another device, say a smartphone. The watch collects data from a patient, and the collected data is sent to the program that analyses the data. The graphics of the data collected can also be visualized on the smartphone. In the case of an emergency, this program can call the emergency ambulance service by giving the patient's current position while sending the analyzed data to the medical doctor, and the results can alarm the emergency contacts. That is not rocket science, after all we have plenty of such systems. In the Semantic

Web project,[3] Berners-Lee, Hendler, and Lassila (2001, p. 43) predicted that "It is not hard to imagine your Web-enabled microwave oven consulting the frozen-food manufacturer's Web site for optimal cooking parameters." The microwave can withdraw such data, and the instructions are translated into the microwave's understanding. The machine 'knows' how to cook whatever frozen food is chosen. However, we could not have reached the Semantic Web yet, as we will see in Web 4.0. That, of course, affects the realization of IoT to a degree. So, we must find out IoT's semantic aspect in *industry*.

2.2.3 Smart Factories

Smart factories, although they are often said to be a component of Industry 4.0, are the realization of Industry 4.0, which mainly depends on CPS and IoT. CPS provide a network of communication between physical and cyber machines and enable decentralized decisions. On the other hand, IoT provides a giant network for various CPS and other physical and virtual devices. Thus, the machines[4] can cooperate in real-time. As a consequence, smart factories indicate value chain organizations,[5] which means the very aim of Industry 4.0 is to create value/advantages in the areas of design, production, marketing, transporting, and service. In that case, Industry 4.0 needs to provide various personalized production and servicing models by enabling and analyzing customer/consumer interactions.

The most significant paradigm shift from a centrally controlled industry to a decentralized production and distribution industry can occur only when human-machine communication, multi-agent level planning, decentralized decisions are taken at concise notice, and data generated and transmitted across companies are held on the Internet. Thus, we can deduce that realizing Industry 4.0 faces

[3] See section From 1.0 to 3.0: Industry, Science, Web.

[4] Once again, we speak of both physical and virtual. In CPS, there are physical machines that communicate with their cyber twins, yet over the IoT, cyber machines of CPS can communicate, as well as software can connect with cyber machines.

[5] Cf. Porter (2011).

the problems of Big Data that we spoke of several times. The data traffic between the collection of machines, either of CPS or IoT, must be manageable in terms that machines can 'understand' the signals from a machine, an error report from software, human inquiries, customer demands, and other potential causes in the flow of manufacturing. The role of IoT is enabling a myriad of devices of CPS and other machines –both virtual and physical– to be gathered in a network that communicates with each other. If IoT is founded on exchanging data or information transfer –and it is– then it is evident that structured data limit both CPS and IoT; in other words, they cannot utilize Big Data. Consequently, we need to find novel ways to use unstructured data, and the information-driven industry will be in the future.

2.2.4 An illustration of Industry 4.0

Lastly, we would like to tell a story that can illustrate an experience of Industry 4.0. Once upon a time, in a dark factory, DT1, a robotic arm breaks down; however, the signals have already alerted Machine-A, the machine responsible for a product line, that that specific robotic arm was about to be defunct. Machine-A decides that the work of the production line cannot be completed in due time, then it sends a message stating that there would be one day delay to Machine B in the distribution section. That Machine-B rearranges the logistics schedule, and it sends a message saying that the autonomous trucks will take the products from DT1 one day later. Meanwhile, Machine-A orders a new robotic arm. Machine-C, in another dark factory, DT2, producing robotic arms, receives the message and checks the reports from the warehouse. Machine-C sends a message to the autonomous vehicle, α15616. The message tells that the robotic arm, AA-411-1511-REA, would be picked and delivered to Section H1 in DT1. However, the reports from the warehouse suggest that the sales of AA-411-1511-REA are increasing. Then, Machine-C urges both the machines and the human designers and engineers responsible for the design and production of AA-411-1511-REA. It asks them to discover why AA-411-1511-REA breaks down so often and what should be done to make them more durable. For the sake of saving time, we stop the story here.

The script tells that the machine manages all the decisions in the supply-production-distribution processes. In a nutshell, the machine conveys all the mechanisms and management.

2.3 Science 4.0

Previously, we indicated a belief that the widespread use of Big Data and the prowess of new technologies in data analyses have shifted science into its fourth era. In this part, we want to survey further and ask whether or not Big Data has revolutionized science at all.[6]

Computer-aided science and digital science, which are often used to name the third phase of science, highlight the involvement of machines in scientific practices. Many labels have been used to describe the last phase of science: eScience, data-driven science, and data-intensive science. These names stress the data deluge that has transformed science. Then, could we state that the data deluge from a new science- viz., machine science- or do all of these labels refer to the same kind of scientific practice? The answers to these questions will be addressed later; now, we want to emphasize that machines' doing science is not a new concept. Regarded as the first programmer, Ada Lovelace dreamt of machine-made science centuries ago: Machine-science was an agenda among scientists, and scientific reasoning done by the machine lies at the heart of it (Boden, 2016, pp. 7–8). Turing's seminal question of whether the machine can think subsumes the potentiality of machine-science; any affirmative answer to this question then yields machine-made science. Although predating this question by five years, Bush ([1945] 1979, p. 40) speculates the following.

> [...] we can enormously extend the record [data]; yet even in its present bulk we can hardly consult it. This is a much larger matter than merely the extraction of data for the purposes of scientific research; it involves the entire process by which man profits by his inheritance of acquired knowledge. The prime action of use is selection, and here we are halting indeed. There may be millions of fine thoughts, and the account of the experience on which they are based, all encased

[6] Some excerpts of this part is published in Yargan (2020a) with minor modifications.

> within stone walls of acceptable architectural form; but if the scholar can get at only one a week by diligent search, his syntheses are not likely to keep up with the current scene.

Bush mentions the extent of *bulk* of data and the process of *selection* to highlight that the cumulative growth of data is inevitable and that he calls for something capable of pursuing the selection process for scholars. That thing Bush had imagined was a machine called *Memex*, a hypothetical device that allows users to navigate between books, records, and communication. These data were to be stored in the device by the user. It would be fair to state that Bush was worried about the amount of data in the Memex, and so should we be, as well. From Bush's vision of the Memex until today, we have faced the problem of accessing the data we request, the reason being, as Bush mentions, is selection. Selection requires understanding; otherwise, how can the relations be set between what is in the Memex and what the user inquires? A simple inquiry could take years if it were not for the machine that can conduct extensive searching. Today, there are search engines, and the successful ones list the related data based on the inquiry's keywords. However, even today, we have to click links to check whether the link includes a genuine answer to the inquiry. Therefore, considering the bulk of the data, a simple search process cannot help scientists find relevant documents that they are looking for. A simple search in PubMed, a life sciences database, requires its users to select, which is impossible unless the user is lucky enough that the document they are looking for is on the first page. In a nutshell, searching narrows the search space; however, it is not that significant in a data deluge. Amid vast volumes of data, even the selection of the required document is hardly feasible. Therefore, we need to find ways to reach the document we need without a diligent search. That means there should be an infrastructure in the silos, where searches are conveyed through the knowledge in the document rather than through the documents' metadata. That is another way of stating the requirement that the machine is supposed to answer queries automatically.

2.3.1 Statistics, Explanations, and Machine Learning

In the course of the data deluge, data scientists have been studying ways to extract data that can be used for solving the problems at hand. Tools are tailored for specific problem sets; for instance, for image recognition, there are several models that are used, or for text mining, there are different models that can be employed. These tools, developed and employed by data scientists, are mainly statistical or probabilistic models. Statistics is the discipline that aims at finding relations between data sets and hypotheses. The data sets are codified and structured empirical facts, and hypotheses are general statements about the target system, where the statements are expressed by probability distributions over the data (Romeijn, 2017). Probability, on the other hand, studies the likelihood or chance of the occurrence of events in a quantitative manner (Demey, Kooi, & Sack, 2019). In other words, probabilistic models in science deal with uncertain situations. Statistics is more about the past, the probability is more about the future, yet both are to *assist* the researchers in explaining the scientific problems quantitatively. Thus, data scientists use statistics and probability to build models to *evaluate* hypotheses in light of the structured data. One caveat must be added: in science, explanation refers to many models, such as the deductive-nomological model, the statistical relevance model, and the causal mechanical model (Woodward, 2017). Some of these use statistics and probability as valid explanation methods.[7] These models are cordially welcome in all phases of science. However, the mentioned methods that deal with Big Data use statistics and probabilistic models as if they were *the only* scientific method that should be used. The crucial difference between the third and the alleged fourth phase is that science uses these models as *a part* of the scientific process. On the other hand, the alleged models that deal with Big Data use statistics and probability as the *only* explanatory method. In other words, data analytics unveil the answers to all scientific questions in

[7] Woodward (2017) mentions that in some explanation theories, causation is identified with explanation, viz., explanation is subsumed by causation (Cf. pp. 79ff). We favor the idea that explanation theories should embrace both causal and non-causal explanations. For instance, functions are counted as explanations primarily used in biological sciences.

Big Data (Cf. Anderson, 2008; Mayer-Schoenberger & Cukier, 2013). Pearl and Mackenzie (2018, p. 13) argue that data by itself is "dumb," and it can only tell correlations and associations but can never tell us why; this is the reason why we should dissuade ourselves from the charm of the so-called superhuman discoveries of data analytics.[8] Pigliucci (2009) defends the same idea abrasively, where he attacks the notorious article of Anderson. When Anderson argues that science suffices the correlations employing the data deluge, says Pigliucci, he commits an error that the very nature of science is not finding patterns but finding explanations for those patterns. No matter how powerful machine analytics are, or the cloud technology provides superfast analyses, any scientific practice must contain the formulation and testing of hypotheses (Cf. Pearl & Mackenzie, 2018, Chapter 10). Considering this, science must be accountable; numbers say nothing about the nature of phenomena.

Another challenge for those who support that statistical and probabilistic models for science is satisfactory in Big Data's era comes from Marcus (2018). Given its due in speech recognition, image recognition, and some other classification problems, Marcus challenges deep learning – the deified model for every task, a statistical technique that uses neural networks with multiple layers to classify patterns- from several aspects. Firstly, deep learning cannot learn abstractions through explicit, verbal definitions; moreover, it needs to crunch millions or billions of data to provide satisfying works (p. 7). Furthermore, the patterns that deep learning extracts are too shallow, and deep learning cannot deal with the hierarchical structure of the language (p. 9). Besides, such models cannot relate the background knowledge or open-ended inferences (pp. 10–12). Above all, deep learning models are black boxes, and correlation-based makes them unreliable tools at all (p. 11; p. 13). As of 2022, no deep learning model copes with these aspects.

[8] Pearl and Mackenzie (2018) offer Bayesian networks as the mathematical foundation of causal inference in science. On the other hand, Pearl admits that Bayesian networks cannot understand causes and effects. Pearl's Causal Revolution model will not provide any causality model in Big Data because it lacks semantic aspects.

Now, let us discuss Marcus' objections against deep learning models within scientific practice. If the machine cannot learn abstraction, then how can it serve as an inference system? If we assume that the machine can learn abstractions when there are millions of data, could we conclude that such a volume of data actually exists? As Leonelli (2014) stresses, structured or usable data is too limited in some fields because data are organized differently in different research, and data curation is labor-intensive expansive work. For instance, Leonelli insists that there is nothing like a data deluge in some branches of biology. As Kahneman (2011, pp. 109–118) elucidates, deciding on the sample size is troublesome in statistics, researchers may not be confident in deriving intuition or results from their sample space. If this is the case, how can we decide how many petabytes of structured data, if the data exist, are enough to be used in scientific studies? Moreover, abstraction also demands determining hierarchical structures, which is crucial in meaning extraction; otherwise, how can the machine establish complex relations? Suppose deep learning algorithms can only determine the flaw in relational structures. In that case, the recursive feature of the language cannot be represented by deep learning algorithms, which leads to incorrect or no inferences.

Additionally, reasoning mandates background information and open-ended inferences. For instance, when we say that all humans are mortal and that Laura is a scientist, we know that a scientist is a human, and therefore Laura is mortal. This inference cannot be made by deep learning algorithms since if the scientists' being human is not provided explicitly; *even* if it is provided, it cannot make such an inference. One may say that deep learning is about classification and may even criticize us for speaking about deduction. This relevant claim can be defended by the fact that deep learning cannot classify abstract concepts (Cf. Marcus, 2018, p. 7), and that the non-hierarchical structure does not allow property- and relation-transfers. In order for the machine to employ deduction, algorithms should construct classifications, which in turn enables the subsumption relation to function so that the child class can subsume attributes of the parent class. That is to say, deep learning algorithms cannot classify but cluster

concepts. Hence, the fabulous deep learning models can neither represent nor generalize the rich structure of the world in a reliable manner.

Above all, one of the most critical aspects of doing science is communication, and since deep learning models do not have a transparent nature, they cannot give an account of results. For instance, scientists need to know how the machine comes to such a solution in medical diagnoses. The machine must scrutinize the verbal clarifications of the studies and the results if they are to be part of a science team in the era of Big Data. That is to say, the black-box feature of the machine should be eliminated. Moreover, potential biases cannot be detected in such opacity. All that said, an explanation cannot be endowed with deep learning (Pearl & Mackenzie, 2018). Not providing representations of causality puts deep learning in a dangerous position. Albeit their victorious results in perception classifications, deep learning, and other statistical and probabilistic models used in Big Data analyses cannot give rise to a shift in science.

Further analysis can be surveyed in statistics. Shmueli et al. (2010) states that explanatory and predictive statistical models are the most commonly used modeling in sciences for the purpose of theory building and testing/generating theories. They are not used to reveal the truth behind the operationalization but are used to *assist* researchers. Shmueli also stresses the fact that neither can predictive models provide explanations nor do explanatory models predict. The main point of the distinction between these models is due to their having different practical implications. Thereby, even in statistics, we cannot and should not expect predictive models to shed light on statistical explanations; those are *just* used for guiding researchers about the issue, so how come can we generalize the results of predictive models as solid truths of reality? She also highlights that notions of "explaining" and "predicting" have been under debate in the philosophy of science for many years (pp. 292–293), at different granularities; these are to point out various aspects of phenomena, even in science. We often fall into error in equalizing these two notions due to our perplexity on the notion of theoretical prediction. A theoretical prediction is an assertion that is driven by a causal theory. However, the whole story we are discussing so far is that accurate predictions often cannot be produced from causal-explanatory models.

To sum up, Jim Gray distinguishes computational science and data-intensive science (Hey, Tansley, & Tolle, 2009, pp. xix–xx). The former refers to science as conveying simulations of complex phenomena, which is the computational branch of science. In contrast, the latter is about scientific explorations, where the data is captured and/or generated and then processed by the machine. Namely, in computational science, the machine extends meaningful data for scientists; in data-intensive science, the machine also contributes to data analysis. Considering Gray's distinction on the subject matter, we could claim that the machine has become a crucial part of scientific discovery, which is undoubtedly revolutionary for scientific endeavors. We admit the fact that the transition from, say, computational ecology to eco-informatics is a significant shift, yet, on the other hand, we do not give enough credit to the fact that "almost everything about science is changing because of the impact of information technology" (p. xxx). As Gray mentions that scientists must codify their data and findings for information exchange with other scientists (p. xx), standardization of representation is a must. Indeed, Gray is aware that such standardization is cumbersome, and working on semantics *may* take infinite time (p. xxix). In this case, our understanding is that if computational ecology is data-intensive and technologically advanced, it is defined as eco-informatics, which is considered within the boundaries of data analytics. Unless semantically enriched standardization is introduced into computational scientific practice, it would not be easy to contend that a fourth paradigm is emerging in science. We admit that scientists must develop more skills to deal with data; accordingly, they must use certain specific software efficiently. From a scientific perspective, such a requirement is a crystal-clear change; however, we insist that doing science with the machine has not dramatically transformed how we produce scientific knowledge.

2.3.2 Big Data and Machine Science

Current improvement in the technology of experimentation and measurement yields a vast amount of data, which is often called the data deluge. Researchers are looking for new methodologies to digest such a vast amount of data in order to offer a novel framework for doing science. This framework is called eScience, or

data-driven science. It is undeniable that this data deluge forms the future of all sciences, and accordingly, scientists must acquire new data analytical skills. On the other hand, the enhancements in computational power and algorithms and the data, which have never been that immense, are just improvements in From 1.0 to 3.0: Industry, Science, Web. Two caveats were mentioned in this part: Firstly, neither artificial neural networks nor Bayesian inference nor machine learning algorithms, namely, no data analytics tools, can give an account of their results. Explanation in science is vital: No explanation translates to no scientific claim. It is only when the machine can answer the question "why?" can we trust their results as scientific discoveries. As we have stressed, statistical models are used in science to help researchers develop their theories. If we aim at creating the machine that provides recommendations to researchers –humans who cannot read all documents both directly and indirectly related to the research at hand– then the machine could understand the content provided by the researchers. That will only be possible when scientific Big Data has incorporated semantic aspects in its analysis methods. That brings us to the second part: We need to deal with Big Data semantically. Although computer simulations and data from devices provide structured data, and so many taxonomies and ontologies are used in specific areas of science, structuring scientific Big Data is a tall order.

Semantic work done so far is labor-intensive. It is hard to find scientific standardization since researchers are free to conceptualize their works. Semantic standardization is often done by constructing ontologies. When each laboratory or branch of science has its own ontology, then there must be other ontologies to bridge together two different ontologies representing the same domain and, in turn, another to bridge a related ontology. Whether it is *ad infinitum* or not, data curation in Big Data is hardly possible. If the machine were to provide recommendations to researchers, how can they relate scientific documents? There are millions of published papers in science, and it is impossible for us humans to read and analyze them. We have to find ways to make use of data in darkness. Unless we find ways to standardize data, Big Data will never be able to revolutionize science. In order for this to happen, the machine must be capable of data curation; it must automatically be able to represent unstructured data

in a structured manner. In addition, the machine must be able to cope with open-world systems since computer simulations, experiments, and scientific documents reflect one aspect of science. It would be an open system when all of these are brought together. Only when we can finally achieve these points will we actualize the dream of Ada Lovelace: Machine Science.

2.4 Web 4.0

Berners-Lee et al. (2001) portray the Semantic Web as the technology that can manipulate the contents on the Web meaningfully and automatically. Accordingly, it is possible by the machine's accessing the *structured* collections of data and sets of inference rules; this way, automated reasoning can occur by applying data to rules. Consequently, the machine could read the contents on the Web, make relevance and cluster the pages, and then execute its intelligence on making inquiries and generating contents. On the other hand, as Boden (2016) lays bare the fact that the Semantic Web is a tall order, let alone being state of the art. In this part, we discuss the current status of the Web, the search-inquiry dichotomy in the Web and the notion of content classification and generation, and the limitations that encumber with the Semantic Web. Finally, we will speak of the future of the Web as Web 4.0.

2.4.1 Big Data and the Web

The biggest effect of Big Data is on the Web. Daily-created data on the Web exceeds petabytes: Per day, billions of photographs and videos are uploaded; billions of comments are posted; millions of webpages are created or updated; thousands of documents are uploaded. Search engines help to find the *related* webpages based on the keywords in the inquiry. The results, on the other hand, may be millions of pages! As of July 18th, 2019, our Google search on the meaning of life resulted in about 1,360,000,000 pages and took just 0.67 seconds. Such a huge number of pages can be listed in less than a second; alas, a lifetime is insufficient to read even a quarter of these pages. On the other hand, search

engines of the day are skillful: Based on our previous web searches and webpage visits, they can map our areas of interest, and then they provide a prioritized and personalized list with respect to our traces. Suppose that one often searches about Douglas Adams and then decides to search for the meaning of life. On the first page, contents about the number *42* and *The Hitchhiker's Guide to the Galaxy* inevitably occur. Let us consider the same issue from a specialized website. On the PhilPapers,[9] a comprehensive index and bibliography of philosophical works, we searched "formal ontology + computation" as of July 18th, 2019; there were four results, which refer to books or articles. The keywords in this search are "formal," "ontology," and "computation"; the results contained these words either in the title, in the main text, or in the title of the journal. Although what we asked was the pages containing "formal ontology" and "computation," there was a result containing "formal computation ." The results were then just listed upon the case that the documents included the keywords. This was not what we were looking for; there should have been at least a link directing content that discusses both formal ontology and computation, or there should be a remark that there was no such book, article, or journal. The links to books, articles, journals, and blogs are categorized in PhilPapers. Such categorization makes the searches manageable; nevertheless, it was not the content but the titles were categorized, yet we want to do searches about the content. For this reason, it is incorrect to introduce it as a database. In a genuine database, on the other hand, data are ordered and labeled, upon which we can inquire. For instance, bank account reports are ordered and labeled: when transactions occurred, how much money was sent on a specific day, and when the money was withdrawn can quickly be answered. On the other hand, when the report was written as a text, then the machine cannot answer those questions unless the whole story is structured as in Web 3.0. The amount of data is increasing with accelerated velocity and immense variety, which is hard to categorize, if not impossible. That means search engines are doomed to list millions of pages. At the bottom, categorizing, labeling, and ordering the content is not unique to

[9] `https://philpapers.org`

the Web in the era of Big Data. The data deluge reminds us that structuring the data in the Web is the first issue we need to deal with.

The grueling impact of Big Data on the Web explicitly manifests itself when summary, insight, and analysis of thousands of webpages have become crucial in areas such as economy, politics, or markets, which need to be analyzed almost in real-time. Investors, for instance, follow Bloomberg,[10] which is a news agency aiming to provide business-related news, including real-time and historical price data, financials data, trading news, research, and expert reports. The real merit of Bloomberg, however, is the Bloomberg Terminal (BT), or Bloomberg Professional Service, which is software that provides real-time data on markets; breaking news from various categories, which affect the market -politics, currencies, or bonds to name some-; data analytics, and execution capabilities all-in-one. Financial professionals can track and analyze the relevant breaking news that is accessible and reliable. Moreover, the news professionals deliver their analyses on the news and can suggest the best possible decisions. All these make Bloomberg a trusted source for finance experts. Besides its Terminal, Bloomberg embraces a television channel that broadcasts 24-hour global business and financial news; a radio that broadcast news and talks; a webpage that hosts articles, videos, and numbers; and social media accounts. It is not only the data created outside Bloomberg but also the data Bloomberg has created matters. What actually Bloomberg does is send the related documents to BT users and leave the real analyses to be taken by the users. The end-users are financial professionals who are the ones to take steps after the data provided by BT. Therefore, it is legitimate to say that BT does not make the data analyses; rather, it filters the texts, videos, and transcripts and then sends the relevant ones to the correspondents. In the data deluge, how can an expert read all those texts, watch the related up-to-the-minute news, and act accordingly? Finance, above the most, is an activity that requires an up-to-the-minute move, which means instant analyses and interpretations are vital.

[10] `https://www.bloomberg.com`

2.4.2 An Alleged Solution: Deep Learning

Now, let us investigate a popular tool that is believed to cluster web contents and summarise them. In May 2020, OpenAI,[11] an AI research and deployment company,[12] announced the largest language model ever created. Called *Generative Pre-trained Transformer 3* (GPT-3), the model has triumphed over many NLP tasks, and benchmarks (Brown et al., 2020). GPT-3 can be applied for all NLP tasks with few-shot demonstrations specified via text interaction between the model-user and the model. This autoregressive language model with 175 billion parameters[13] has gained remarkable accomplishments in translation, generating computer codes, question-answering, mimicking literary styles, using a novel word in a sentence, or generating news articles that human evaluators had a hard time identifying them (Brown et al., 2020). The novelty of GPT-3 lies in that the model is fed *raw data*, then it generates structures based on the raw data and with few-shot demonstrations. For instance, one can give two words with their Turkish equivalences, such as "hope – umut" and "love – sevgi."[14] When the user writes "joy" on the console, GPT-3 acts like a translator and returns "neşe."[15] Or when it is fed scientific data, it generates text scientific-like response. Can such a powerful tool be Web 4.0? Maybe not GPT-3, but GPT-n, where $n > 6$, will be a query engine over the Web. GPT-n will be able to generate a screenplay of *İnce Memed* in Tarantino's directing style. That is to say, the future of the Web hinges on deep learning-based language models with gazillions of parameters. Nevertheless, we doubt it.[16]

Deep-learning-based language models cannot pave the way for Web 4.0 for two

[11] https://openai.com

[12] The company aims that artificial general intelligence benefits all of humanity by building safe and beneficial machine intelligence.

[13] That is a lot.

[14] This is a two-shot demonstration.

[15] For a variety of examples, visit `https://gpt3examples.com`

[16] In the scope of this work, we will only mention Web 4.0-related issues. For social concerns of GPT-3, see Brown et al. (2020), pp. 34–39.

main reasons. Firstly, the impressive achievements of deep learning amount to curve fitting. That is, the deep learning algorithms produce almost-exact parameters that suit the data best, which is nothing but identifying associations between the nodes in the neural network. For instance, deep-learning language models cause disaster when reasoning by association is replaced with causal reasoning. The syntax they perform is unquestionably fluent, yet relating the words to real life, namely providing a degree of semantics, is out of the question. That is not a reliable representation of reality. As the manipulated data lack comprehension, such models cannot make for trustworthy Web 4.0. Therefore, they are unreliable interpreters of reality (Marcus & Davis, 2020).

Secondly, we cannot back up interpretations of the models. Specific to GPT-3, the model seems to produce new contexts, summarize, and answer questions ingeniously by manipulating heterogeneous unstructured petabytes of data. Nevertheless, the Web is full of various contents with intolerable contradictions. Deep-learning-based language models do not learn about reality; instead, they learn about how content producers use words in relation to other words (Marcus & Davis, 2020). Since such language models are grounded in frequencies of sets of strings in the texts on the Web, outnumbered misleading statements can be taken as the truth of the world. For instance, drinking disinfectants to treat COVID-19 disease can be ordered by a deep-learning-based language model; moreover, the confidence of the recommendation increases when an authority utters it.[17] Although Brown et al. (2020, p. 34) utter lamely that decisions of GPT-3 are not interpretable, *any* language model based on deep learning is a black box. We cannot know how the conclusions are derived from a system where meaning is distributed.[18]

[17] See Trump's suggestion of using disinfectants against coronavirus `https://www.youtube.com/watch?v=HKCe8Pq_RUA`, and its news on The Guardian: https://www.theguardian.com/us-news/2020/apr/24/trump-disinfectant-bleach-coronavirus-claims-reaction.

[18] Bender and Koller (2020) acknowledge that deep-learning-based language models can capture semantic similarity but stress that semantic similarity cannot lead to the actual meaning. They show that meaning cannot be learned from a form of language alone. The defenders of the neural network architectures, however, insist that the machine can *explain* what led to the conclusion.

2.4.3 An Automation of Classification of Webpages

The main puzzlement in Web 4.0 concerning the machine's answering inquiries and generating content is that most of the content on the Web is unstructured, which is discussed at length in the appendix section From 1.0 to 3.0: Industry, Science, Web. That is not only crucial for Web 4.0 but also for Industry 4.0 and Science 4.0 since data related to these fields are also embedded in the Web. Thus, the Web must behave just like an online agency. According to the market, an agency's webpage categorizes its products and services, among which one can make a choice. For instance, consider that one plans a vacation. There are thousands of locations and millions of commercial lodgings. A successful travel agency's webpage categorizes types of tourism (ecotourism, cultural tourism, wildlife tourism, and alike), destinations (continents, counties, cities, sites, and alike), commercial lodgings (hotels, B&Bs, condos, and alike), and other categories. So, the person who wants to travel only looks at the results narrowed down, afterward books the most appropriate one that fits their preferences. Thus, Web 4.0 must be capable of classifying all the available content on the Web according to the queries. That is Web 4.0 structures the contents according to the context of each query. For instance, "meaning of life" is a query in Web 4.0. The machine reaches all the related web sources and classifies them so the user can choose the most convenient category. The automated classification of the webpages narrows down the search space for the user. For instance, a query page can offer the following classifications: definition of life, the meaning of life in philosophy, meaning of life in religion, meaning of life in philosophy, meaning of life in biology, the meaning of life in philosophy, meaning of life in psychology, "meaning of life" as an artifact. When "meaning of life in religion" is chosen, then the machine can even narrow down the search space and offers some subcategories: Creator-centered; nature-centered, individual-centered, Zeitgeist-centered, and so on. Then, the user can click on the subcategories and see the "to the point" webpages. Hence, Web 4.0 enables classifications that narrow down the search space and customize the query so that users do not need to skim all the pages.

Let us give a vivid example. Suppose that one wants to learn that Ankara is bigger than what. The machine categorizes all the webpages that include information about the size of Ankara. Then, it can prepare a list that contains some categories and their subcategories and the answers, such in Table 2.1, where the counties in Europe whose population is lesser than Ankara are listed.[19]

Table 2.1: An example of Web 4.0 categorization of a query

Ankara is bigger than ...	
landscape (cities)	Europe ‖ Asia ‖ South America ‖ ...
landscape (countries)	Europe ‖ Asia ‖ South America ‖ ...
population (cities)	Europe ‖ Asia ‖ South America ‖ ...
population (countries)	*Europe* ‖ Asia ‖ South America ‖ ... Finland Slovakia Norway Ireland Croatia Moldova ⋮ Holy See
⋮	⋮

To conclude, ontologies, as Berners-Lee et al. (2001) offer as an integrated technology for coping with semantic issues, are not capable of hosting unstructured entities. The future of the Web, or Web 4.0, is the Web as a giant database that delivers trustworthy content. The machine, then, is supposed to read all the related sources, interpret them, and classify them according to the search. The next feature of Web 4.0 can be answering queries, as well. Such a trademark of Web 4.0 will generate content based on a query, not based on a search. A query process requires logical entailment, and so as a content generation. There may be confusion with content or article generator tools, which create unique texts from the existing content on the Web. A user provides some keywords, then assigns the text type -such as article, essay, comment, posts, research paper, blog content, website content-, and the article settings -number of paragraphs, sentences, and words-, and may choose the possible sources. Such software creates

[19] The information is taken from `http://www.ankara.gov.tr` and `https://www.worldometers.info`, on February 14, 2022.

new content by rewriting, shuffling, paraphrasing, and exchanging synonyms from existing sources.[20] The alleged *genuine* content may vary from Twitter posts to a full-fledged article at length. However, Web 4.0 must be able to cluster the existing contents in the webpages first, and then context generation on the clustered webpages can be done.

2.5 Teleological Convergence

Since the conceptual construction of *Calculus Ratiocinator*, machines have been developed to take over the intellectual activities of humans, such as solving complex algebraic problems, recognizing patterns, and answering queries. To employ new intellectual tasks to machines, we need to detect the common requisitions to realize 4.0 versions of Industry, Science, and Web.

A recap of the previous parts is the following. The future of Industry requires autonomous factories, in which the machine automatizes each stage of good productions. For this to happen, there needs to be interoperability in human-machine communication and decentralized intelligence in information-intensive manufacturing. In other words, the machine does operate in an open system as an autonomous agent. In Science 4.0, the machine will be both knowledge sources and agents determining the produced knowledge's status. The future of Science demands that the machine contributes to scientific knowledge production by making recommendations to scientists, answering queries, and explaining the patterns as an autonomous workfellow. The future of the Web, on the other hand, presupposes the machine to become content managers and producers. As the former, the machine will answer the natural language queries in any subject; as the latter, the machine will generate content for a task from the Web's voluminous information. However, the task prerequisite is the machine's ability to classify the webpages for a given search/query. To wit, the Web will be a giant knowledgebase upon which the machine will classify the webpages,

[20] The ethical and commercial issues about these tools are not in the scope of this work.

and then make inferences and generate content for a given topic. Accordingly, we can conclude that the machine is assigned a new role in all the 4.0 versions of ISW: an *actual agent*.

The machine is an 'agent' in the sense of machine intelligence: an autonomous entity that acts purposefully to achieve a task in a context or, say, in an environment. They are 'actual' or 'active' in terms of their involvement: Without them, the mentioned tasks can never be achieved. Therefore, the realization of 4.0 versions of ISW is stipulated in an autonomous explanatory machine: the machine that makes decisions, generates contents, makes recommendations and provides reasons for their judgments under uncertainty due to incompleteness and/or incorrectness of the representation of reality. Hence, we need to figure out the requirements that meet the requisition of the machine's becoming an active agent.

Present-day machine is far from being an agent. They are just expert systems constructed with structured data of a specific domain for a particular task and/or statistical models using structured and/or unstructured data. Decisions/recommendations of the machine are valid only in the domain represented or trained for the machine. That is, the produced results can be inapplicable to another domain, and/or the representations can be insufficient for the other. However, all the 4.0 versions of ISW require that the machine makes decisions[21]/recommendations from Big Data, which is not limited to a specific domain. Put differently, the machine's new role empowers them in processing Big Data, which embodies decentralized structures in open-world aspects. In this direction, the first requirement for the machine's becoming an autonomous decision-maker is the machine's operating in the open world, which refers to the incomplete real-world representations that contain implicit knowledge. For instance, a scientific paper produced by a laboratory reflects the close-world of that laboratory; on the other hand, a collection of scientific papers reflects an aggregation of those close-worlds that can legitimately be considered an open-

[21] We are taking content generation as decision-making because the content is produced upon some purposeful selections and arranged to answer a specific question or query.

world, to which we can make complex queries. Being directly related to the first requirement, the second requirement is that the machine as an agent can cope with the complex dynamic environment consisting of various contexts. That means the new role of the machine strives for a decentralized system. In centralized systems, as in expert systems, a specific environment is represented; to use the same environment for a different task, another representation is usually required. However, if there will be ISW 4.0s, machines in the smart factories have to derive decisions from various environments; the machine has to generate context from the differently tagged documents on the Web;[22] the machine has to communicate automatically between various scientific works of different aims. Hence, if the machine should be an autonomous open-world reasoner and, of course, explainer, the representations that the machine manipulates must be in a decentralized system. A prompt objection may be on the way: if the very nature of the Big Data is unstructured, then how come can an automated system make decisions and recommendations, provide explanations, or generate content with unstructured data with which we cannot build a knowledge processing system?

2.6 The Role of Data

As mentioned above, machine-learning systems cannot create a learning machine: pattern learning is not the same as scientific discovery. Nor can statistical models explain the results: the machine cannot state the reasons for their conclusion systematically.[23] Then, machine-learning models cannot be taken as active agents. Moreover, as highlighted several times, structuring Big Data is a tall order, if not impossible. These methods are incapable of integrating data from various sources, learning and operating in batch and real-time, and being accountable for the generated results. In a nutshell, an active agent must have the ability to learn and adapt as it makes decisions, suggests recommendations,

[22] A fundamental feature of Web 3.0 is the content generators' ability to tag the entities as they like.

[23] Once again, explanatory models are not designed to predict the results, and no model combines both. See section Science 4.0.

provides explanations, and creates content. Such an autonomous being then must *understand* its wild environment, which is Big Data.

Data is a concept encountered in every field and is not that difficult to define since data is a representation no matter what. In its simplest terms, a representation is a kind of notation standing on behalf of an entity in a particular way. Phenomena represented in the machine is possible through datafication, a process through which the machine can manipulate and/or analyze whatever is recorded as a representation. Datafication, in other words, is "depicturing" the world into the machine. To what extent are we successful at providing these depictions that empower the prowess of machine intelligence; accordingly, to what extent can we represent knowledge indicating the prowess of the machine's knowledge processing? In this respect, we claim that the focus should be on data; otherwise, we had already had some glimpses of the realization of any of ISW 4.0 versions. Because today's biggest problem is solving how unstructured data can be processed together with its meaning. Data, tamed in all distinct stages in ISW, is in the wild now: none of the previously-used representations of reality can be employed in 4.0-versions, nor can we speak of the machine operating statistical models to understand data. So, if we want to represent knowledge in ISW 4.0, then we need to datafy the semantic components of an entity, if there are any, even in the wild nature of Big Data.

The following section will investigate the nature of data and discover an appropriate path leading to machine understandability. To this end, it will cover how data is defined in the literature and then define data from a new perspective, which sustains machine-understandability.

2.7 Definitions of Data

The first definition of data is from Ackoff (1989, p. 3): "Data are symbols that represent the properties of objects and events." Objects and events can be symbolized in various sorts; characters, numbers, audio and visual signals, images, and alike. Although this definition is prevalent among information sci-

entists, the desired definition of data must be machine-oriented.[24] Then, we can consult Claude Shannon, a prominent information theorist. He defines data in terms of entropy as patterns of physical symbols/signs. According to this quantitative approach, the semantic aspect is irrelevant (Shannon, 1948, p. 379). Unfortunately, we believe that there must be the requirement of representations of semantic aspects of data for knowledge to be processed. Of several definitions of data, most of the time, it is defined in terms of information as being units/morsels/pieces of information (Cf. Gitelman, 2013). This approach is problematic since labeling something as data or information depends on the perspective.[25] Further, in his work on the foundations of data modeling, George H. Mealy (1967, p. 525) distinguishes three distinct realms in the field of data processing: the real world itself, ideas about it existing in the minds of men, and symbols on paper or some other storage medium. Data can be defined as fragments of a theory of the real world, and data processing juggles representations of these fragments of theory. This definition is close to what we are looking for, yet the semantic aspect of data is not emphasized. At this point, Frické (2015, p. 652) succors by introducing data with a semantic value and in a machine-oriented way.

> Data is anything recordable in *a relational database* in a semantically and pragmatically sound way. The semantics require that the recordings be understood as true or false statements. The pragmatics suggest that we favor recording what seem to be concrete facts (i.e., singular and relatively weak statements) and that interpreted recordings be true statements. [Emphasis added.]

This definition, however, is *a* definition of structured data. Similarly, Mayer-Schoenberger and Cukier (2013, p. 78) define datafication in terms of structured data: "To datafy a phenomenon is to put it in a quantified format so it can be *tabulated* and analysed" [Emphasis added]. Accordingly, data is a description of something that is "recorded, analysed, and reorganized" (ibid). As we stressed

[24] Hence, we ruled out any non-machine-oriented definition of data. In addition, we find the data-information-knowledge-wisdom (DIKW) pyramid misleading.

[25] The main difference between data and information is that the former is self-standing, whereas the latter requires the former for its existence. Consider a timetable. When we use it for extracting specific information, the timetable is data. Because the table is full of texts and numbers gathered some facts based on records about something. On the other hand, it can be information since it is refined data based on analyses and is used for making judgments on facts.

above several times, to grapple with Big Data issues, the crucial challenge is to cope with unstructured data, to which the limitations of Science 4.0, Industry 4.0, and Web 4.0 boil down. On the other hand, this definition can encapsulate the unstructured data as well, one may say. Indeed, we can tabulate unstructured data with analytical tools; yet these tools cannot operate on semantic aspects of datafied entities.

For our purposes, data must have *processable semantic properties*. In light of this, data can be defined as computable semantic parts of the representations of entities. In this respect, it has two indispensable aspects: semantic and computational.

Firstly, the semantic aspect of data denotes that data representation inherits 'knowledge' from the real world. In other words, data are represented with semantic properties along with data values and data types. Secondly, the computational aspect of data denotes that the machine processes data. However, we are not mentioning a set of pixels, which is processed to constitute a picture in the machine. Since machine understandability is our concern, the machine should compute the semantic properties of data. We will revisit these two aspects in Data 4.0.

Having said all these, we define data as follows.

Definition 1 Data *is the essential notion of representation in the machine.*

An eagle-eyed reader may discern that there is no mentioned semantic notion in the definition; however, what inherits semantic properties is the *representations*. All of the data neither be structured nor semantically laded nor in the human-understandable fashion. Thus, our definition encapsulates all sorts of data. In this respect, we claim that there are four categories of data. Data 1.0 is machine data, the most basic category, which refers to any data that is processable in a machine. Data 2.0, the machine information, includes data that is transferable between devices/machines. Data 3.0, machine knowledge, is a data collection in machine-readable form. Lastly, Data 4.0, machine discovery, consists of data with processable semantic properties. This category is the ultimate

representation of phenomena, which paves the way for 4.0 versions of ISW.

Note that the traditional Data-Information-Knowledge-Wisdom pyramid does not apply here. According to this pyramid, there is a gradual construction from data to wisdom. More precisely, the DIKW pyramid offers that the bottom of the pyramid is data that is bare and/or random raw facts; the layer on the data is information that is refined facts gained through understanding the relations between data. Knowledge is the top layer of information that is an accumulation of refined facts gained through understanding the patterns of information. Wisdom, at the top of the pyramid, is the value of knowledge gained through understanding the principles. This pyramid is not only intellectually anthropocentric but also its layers necessarily depend on the bottom layers. On the other hand, Data 1.0 refers *sine qua non* to all data; yet, for instance, Data 3.0 is not an upgrade of Data 2.0. We have to ask what it is to be machine knowledge rather than modify this pyramid for already existing machine processes. On the other hand, we highlight four categories of data that can be labeled according to their representational notions. Data 1.0 is called machine-processable data since data is represented in a way that a machine can process it; Data 2.0 is called machine-transferable data since data is represented for enabling machine communication; Data 3.0 is also called machine-readable data, in which data is represented with labels as in Web 3.0; Data 4.0 is called machine-understandable data since data should be represented such that the machine can process semantic properties. That is, Data 4.0 enables us to build a robust discovery system. After this succinct definitional introduction of data categories, let us issue them one by one.

2.8 Data 1.0

The most basic category of Data is Data 1.0, which refers to **machine-processable data**, viz., machines process the data in a definite way. More precisely,

Definition 2 *Machine Data: A formally structured collection of representations appropriate for processing.*

In the early eighteenth century, paper types with punched holes were used to control a loom in textile machines.[26] Each line of the paper types is punched, which means each hole represents digital data (as punched and not-punched) as instructions for a machine to perform the desired pattern to be woven. This automation of loom operation is often considered the first step in our digitalized world.

Charles Babbage is famous for using punched cards for computational purposes, and thus he is regarded as the pioneer of digital computational machines, viz., modern computers.[27] Babbage adapted the punched cards mechanism to reach his goal, a general-purpose machine, the Analytical Engine. In this machine, which was realized after Babbage's death, the cards were used as input for representing formulae, which specify the arithmetical operations that the machine should perform; and as controlling iteration and conditional branching mechanisms for executing the algorithm. Within the years, these digital computing machines became electro-mechanical (Randell, 2013). For instance, Zuse released Z3 as the world's first general-purpose program-controlled machine (Copeland, 2017).[28] Until the advent of electronic computers and then digital computers, only arithmetical operations, statistical calculations, or complex mathematical problems were represented to be processed by the computers at those times. The brilliant ideas of Turing –the Turing Machine–, and von Neumann –the von Neumann Architecture– gave rise to digital computers and then personal computers.[29] From then on, the number of processable data has excelled: we can represent sounds, images, movements, texts, and others. Like in human cognition, in the machine, different physical phenomena are represented in a different fashion: A music piece is represented in 0s and 1s differently from a text file. Like our specialized sense organs, special programs process different kinds of representations. As long as we develop effective tools to represent phenomena,

[26] See section From 1.0 to 3.0: Industry, Science, Web.

[27] However, Semen Korsakov used punched cards for computation and information storage for the first time in 1832 (Shilov & Silantiev, 2016, p. 85).

[28] Moreover, the scope of usage of these machines was increasing. These calculating devices were utilized as totalisators, ballistic calculators, and cryptanalysis machines.

[29] For more information about the historical background, see Copeland (2017) and Randell (2013).

phenomena can be processable by the machine. Returning to the subject, data represented in the punched cards in digital units are ancestors of Data 1.0.

To sum up, processable data is data that represents a phenomenon, and that representation is executable by a machine in a particular way. Data processing is a manipulation of data by a machine that can convert raw data to machine-readable format (in the sense of syntactic operations), transferring data through CPU and memory to output devices, and formatting or transforming output (Encyclopædia Britannica, n.d.). In this sense, any input from input devices (a figure, a mouse click, a character typed in a word processing document), a file to be run (an MP3, a PDF document, a movie), a program, and even a compiler is data (M. Davis, 2000). Having said that, in principle, any data that is processable by a machine is in the scope of Data 1.0.

2.9 Data 2.0

Data 2.0 refers to **machine-transferable data**. The simplicity of this definition, however, may be misleading. 'Data transferring' and 'transferable data' are used interchangeably, and both have several senses. Data stored in a storage medium, e.g., a punched card, a floppy disc, an external memory, or the cloud, can be transmitted to a memory device, e.g., to another punched card or an SSD card. Data that moves from one place to another without being executed by a program is not in the Data 2.0 format. Neither is the reproduction of data: For instance, burning a DVD, the data in that DVD being *transferred* to the other one, is not in the realm of Data 2.0. Thus, *copying* data is not a process of transferable data. Another consideration of Data 2.0 may be that inputs are transferrable data. It is quite reasonable to think that the first telegrams transmitted the data from a sender to a receiver. It must have been transferable data that telegrams forward those processable data. However, it is trivial that Data 1.0 and Data 2.0 would be the same things. Telegrams, cables, and signals are all doing together, converting mechanical or electrical impulses into another kind of impulse that the receiver can process. Hence, such conversions cannot be regarded as machine-transferable data.

The sender-message-receiver pattern approach, where the message is the information in a transferable format, can be used to understand the machine-transferable data. In general terms, information is useful processed data (Ackoff, 1989), yet the *usefulness* of the processed data for a machine hinges on the program it runs. Shannonian approach to data can be recognized as the identifier of Data 2.0. According to Shannon, information is a sheer quantity, and how much data is transmitted/transferred can be calculated. The intentions of these two definitions differ from ours, although we all want to define *information* from a machine perspective. Let us combine these two perspectives: Ackoff's information needs to be processed, and Shannon's information needs to be transferred. Ackoff's information is by itself data to be processed, viz., is a Data 1.0; Shannon's information is the quantity that remains after the transfer; that is, data is not processed within the machine. If the fundamental aspects of information are processable and transferable, machine information must be data that is both processed and transferable.

The last paragraph echoes with Industry 4.0, alas, whose realization is not possible solely through Data 2.0. Information driven cyber-physical environment of Industry 4.0 can be taken as a collection of dynamic networks. The dynamism emerges due to the interrelatedness/interactions of the components of the environment. In other words, it is more like an environment where the components are in communication. Nevertheless, communication is nothing but a transfer of information. At this point, we can refer to cybernetics, the science of communication –among other things, such as interrelatedness- in the machine.[30] That said, Data 2.0 is more than what Ackoff and Shannon defined because their comprehension of information is linear: a sender sends data to a receiver. This linear process is often illustrated in the input-process-output format. However, cybernetics advocates a connection between outputs and inputs. That is, a feedback mechanism is one of the essential aspects of cybernetics. Consequently, to our understanding, machine-transferable data is closer to understanding information in cybernetics. So, we define Data 2.0 based on the science of communication.

[30] The term 'cybernetics' was first coined by Norbert Wiener, who defines it as above.

Definition 3 *Machine Information: A formally structured collection of representations that flow inter machines.*

2.9.1 Data without Ontology

To take a concrete example, consider a device that records the heart rates of a patient; each record is data. When these data are transferred to other devices, say to an application, they are considered information on those devices. One may argue against it and say that the data that the application is processing are of Data 1.0. We would not object to this claim; nevertheless, such data has a novel feature: communication. With this in mind, Data 1.0 is *processable* data, yet Data 2.0 transferrable *processed* data. Still, a caveat may be necessary here. Machine information needs not to be processed by other devices; rather, it could be processed in the same device but by a different program. Besides, remember that when we call 'machine,' it may refer to virtual machines or programs. This interchangeable usage is widespread in Computer Science. For instance, an intelligent machine needs not to be a robot; instead, it can be a software program.

The machine information has the feature of being data-without-ontology; the feature will be understood better in the following pages. Suffice it to say that the crucial aspect of Data 2.0 is being processable and transferable; there is no specific room for the meaning of data used in machine communication.

2.10 Data 3.0

Reading can be defined with concepts like *looking* -as through which we get the input data-, or *understanding/grasping/discovering* the meaning of *the symbols*. The symbols can be letters, images, or sounds that refer to or designate some meaning our reason can decipher. Let us take 'reading' in the context of reading a book. The letters come up to construct meaningful letter sets, which are words, and the words come together to set a sentence to express meaningful judgment, feeling, or a question. While reading, we make connections between the words

and then between the sentences. Those connections end up with understanding/grasping the meaning of the text as a whole. Reading is a complex cognitive skill, however. Each day, we experience things as categorized phenomena, and those categories have been connected from our birth, if not earlier. Namely, the novel things inevitably are connected to the existing knowledge network.

When the machine is considered, reading is the machine's ability to connect the labeled data. The Semantic Web is a web of data processed by the machine, the data that have been tagged, categorized, and connected so that the machine can 'read' the data. So, Data 3.0, **machine-readable data**, is data that the machine can read. More precisely,

Definition 4 *Machine Knowledge: A formally structured collection of representations appropriate for linking together through their metadata.*

2.10.1 Data with Ontology

Machine knowledge cannot be thought of without an ontology since the machine can categorize and connect data upon tags in a meaningful way. Ontologies are the tools that bring data to semantics. When machine-readable data is regarded as *processable a web of data* of a domain, the semantics of the Web is guaranteed by an appropriate ontology that serves as a backbone.

2.10.2 Machine Knowledge vs. Understanding

One may argue that the machine's ability to read data can result in understanding since the Reasoning Layer in the Semantic Web Stack reveals the implicit from the explicit, where the inference rules are applied only to structured data. Understanding, on the other hand, is the ability to use data from different domains in order to make the implicit explicit. That means understanding is the ability to reach and utilize the data with their all-possible meanings. At the same time, there are two issues to consider. First, the nature of data is mostly unstructured, and second, the machine cannot utilize unstructured data

semantically. That is why we need to introduce a novel kind of data, which is machine-understandable data.

2.11 Data 4.0

Data 4.0 refers to **machine-understandable data**. Let us begin with a caveat. A big mistake is that 'machine-readable data' and 'machine-understandable data' are used interchangeably in the literature. As a wise saying, "Readers are plentiful, thinkers are rare," suggests that reading does not always lead to understanding; thus, the machine should not be an exception. Such a mistake is the opinion that machine understanding is possible by linking the tagged terms. For instance, Tim Berners-Lee (2001, p. 185) defines machine-understandability in the sense that building understanding enables to link 'very many meanings.' He continues that concepts are linked together by "frequent contributions from independent sources" (ibid, p. 187). We have already explained that linking concepts together is not enough to create a meaningful Web. Townsend et al. (2004, p. 3294) speak of machine-understandable data as "highly structured data with varying degrees of curation and annotation" whose characters are not only read by the machine – otherwise, it would only be machine-readable data – but also whose semantics allow autonomous actions. The alleged autonomous actions suggested by Townsend et al. (2004) are the very automated features of the Semantic Web, such as aligning sequences or finding binding sites.[31] To illustrate the confusion between machine-readability and machine-understandability, we need to refer back to the Semantic Web technologies and Data 3.0. In doing so, we can figure out the distinctive features of machine-understandable data.

[31] In the conclusion section, indeed, it is written that "[u]nstructured text is currently [in 2004] impossible to analyze" (p. 3299). In 2022, we are barely getting closer to analyzing such texts.

2.11.1 Things Before: Data with Ontology

Any data is represented in the machine with two fundamental components: data value and data type.[32] When we attain a variable, we name it, like "x," and then assign a value to it, "x = 3".[33] In order machine to understand which operations can be applied on "3", we need to specify its data type. There is another component of data in the Semantic Web: a semantic property. Semantic properties declare the meaning of data in the given context.

In linguistics, *semantic properties* or *features*[34] are defined as the components of meanings of words and sentences. For instance, "human" is a semantic property of "student," and "sense" is a semantic property of "feel." So, semantic properties help to define the semantic field of a word or a set of words; that is, "the semantic properties of words determine what other words they can be combined with" (Fromkin, Rodman, & Hyams, 2018, p. 147). Let us illustrate the semantic properties with a well-known example of Fromkin et al. (2018, p. 152): "The assassin killed Thwacklehurst." We know that *assassin* is a *person* who *murders* someone *important*. A human held the killing action, and Thwacklehurst was not an average citizen. Moreover, we can claim that the assassin performed that act in exchange for money or due to a fanatical adherence. Thus, the semantic field of "Thwacklehurst" includes "human," "important person," "dead," and "murdered."

Fromkin et al. (2018, p. 152) state that "[t]he meaning of all nouns, verbs, adjectives, and adverbs—the content words- and even some of the function words such as *with* and *over* can at least partially be specified by such properties." For instance, the semantic property "location" can be found in many verbs such as "stay," "dwell," "live." A distinction in meaning between the content and function

[32] Data types declare a set of values and operations on these values. In other words, they express elements of a collection, and at the same time, they define a particular access pattern to these elements.

[33] We are not using a specific programming language for the sake of simplicity.

[34] There is no consensus on the meaning of the term 'semantic feature.' Cruse (2017, p. 239) notes that 'semantic features' are also known as *semantic atoms*, *semantic components*, and *semantic markers*.

words is made more delicate by introducing additional semantic properties. For instance, “run” and “walk” share the same semantic property, “motion,” and “run” is distinguished from “walk” by the semantic property “fast.”

In light of this, we explain how data get their semantic properties with respect to layers of the Semantic Web Stack in the following.

`<motion>walk</motion>` This is an XML expression. In plain language, it states that “motion” describes “walk.” Note that there may be other instances of motion like `<motion>squeeze</motion>`.

```
<MyLibrary>
<book ID= ‘‘TR2’’>
<title>Ermiş</title>
<author>Halil Cibran</author>
</book>
<book ID= ‘‘LAT1’’>
<title>Epistulae Morales ad Lucilium</title >
<author>Seneca</author>
</book>
</MyLibrary>
```

This XML expression gives the structured data of one’s library. The first item in the library is labeled as “TR2”, whose title is “Ermiş,” and the author is “Halil Cibran.” The titles of the books in this library are “Ermiş” and “Epistulae Morales ad Lucilium.”

```
<xs:element name="MyLibrary">
<xs:complexType>
<xs:sequence>

<xs:element name="book_title" type="xs:string"/>
<xs:element name="author" type="xs:string"/>

</xs:sequence>
</xs:complexType>
</xs:element>
```

This XMLS expression says that `<xs:element name="MyLibrary">` defines the element called “MyLibrary". `<xs:complexType>` signifies that the “MyLibrary” element is a complex type, because it is a sequence of elements, which is denoted

by <xs:sequence>.

<xs:element name="book_title" type="xs:string"/> and

<xs:element name="author" type="xs:string"> indicate that the elements "book title" and "author" have string data type.

Here we know that the data type "string" is a built-in simple data type from XML Schema, abbreviated as *xs*.

```
<relations>
<author name= ‘‘Halil Cibran’’>
<wrote>Ermiş</wrote>
</author>
</relations>
```

These lines express "Halil Cibran wrote Ermiş" with XML codes.

So far, we have seen how the data of our concern take value, data type, and semantic properties. RDF, RDFS, and OWL are the primary representation languages,[35] and RDF is the foundation since their ultimate role is managing distributed data. Recall also that URIs uniquely identify each resource on the Web, and RDF determines triples in the order of subject, predicate, and object. That means each resource is put in the appropriate position of the triple, and the triples are connected to create a meaningful graph of resources. A graph node can be merged with a node from another graph only when the two have the same URI.[36]

<rdfs:label> Computational Ontology</rdfs:label> expresses that "computational ontology" is a label, which is stated for human understandability. The term "label" is taken from the rdfs namespace, where the label's properties are defined.

geo:lat ‘‘39.925533’’ ^^xsd:float expresses that "39.925533" is of float data type, and it stands for a latitude value. The properties of latitude are defined under the namespace "geo," where "latitude" is shortened as "lat."

[35] For details, see section From 1.0 to 3.0: Industry, Science, Web.

[36] For the sake of simplicity, we are presenting neither URIs nor RDFS/OWL expressions.

`:hasChild owl:inverseOf :hasParent` This means the relation "hasChild" is an inverse relation of "hasParent." In other words, the domain of one relationship is the range of the other one.

`academy:instructor rdf:type foaf:Person.` This RFD triple says that what is "instructor" in the namespace academy is of type "foaf: Person," where "foaf: Person" is an RDFS class defined under the namespace *foaf* (an acronym of Friend of a Friend ontology).

```
instructor:Laura
rdf:type foaf:Person ;
rdfs:label ''Laura Phaenerate''.
```

This RDF resource represents Laura, defined under the namespace instructor, is the type of foaf:Person, and labeled as "Laura Phaenerate" for human readability.

`:isBrotherOf rdfs:subPropertyOf :isSiblingOf` This RDF triple states that all the resources related by "isBrotherOf" are also related by "isSiblingOf."

Further, consider the following XML expression:
`<person><gender>female</gender></person>`. Here, gender is a child element of person. If parent elements were semantic properties, then "female" has semantic properties "gender" and "person." Besides, the same expression can be written as
`<person gender="female"> </person>`. Here, "gender" is defined as an attribute of a person, and "female" is the attribute value. In this case, we can say that gender is an attribute of "person," and "female" is a gender. However, there are no definite rules about when to use elements instead of attributes in the XML; thus, "both examples provide the same information" (W3Schools, n.d.). That is to say, XML tags cannot provide semantic notions; there is no meaningful relation between "person" and "female." On the other hand, RDFS and OWL expressions provide such meaningful derivations. Consider the following schema and the assertion.

```
:Mother rdfs:subClassOf :Woman.
:parentOf rdfs:domain :Mother.
:hasChild owl:equivalentProperty :parentOf.
```

`:Laura :hasChild :Daphne` From the given schema and the assertion, we can infer that Laura is a woman. Because, `::Laura :parentOf` based on `:owl:equivalentProperty`; `::Laura rdf:type :Woman` based on `:rdfs:domain`; and `::Laura rdf:type :Woman` based on `:rdfs:subClassOf`. In plain language, the semantic properties of Laura are parent, mother, and woman, and having a child. All these examples show that semantic properties are constructed in data by *semantic* annotations.

The semantic properties of data become processable by a series of defined interpretations. In the first place, the graph structure of Ontology 3.0 roots in RDF; thus, the semantics of RDF constitutes the fundamental base for all the operations. Then, all the interpretations of RDF and RDFS vocabulary and data types are predefined along with their entailment rules. Further, the axioms and annotations of the domain at hand are specified with an implantation language, for instance, with OWL. The whole representation then becomes processable with respect to its semantic conditions. This procedure will be explained in detail a little later.

Nevertheless, what is machine-understandability? Why do we disagree with the idea that when all the entities are defined in all contexts with all their semantic properties in ontologies, machine-understandability is going to be realized?

2.11.2 Things Should Be: Data within Ontology

The machine-readability roots in annotating semantic properties and processing them in graph structure along with the rules of RDF, RDFS, OWL, and alike, and the constructed knowledgebase. Once everything settles down, the machine just manipulates the data following the inference rules. This means the machine can find explicit facts in the domains and contribute to knowledge production. Is not producing knowledge or finding explicit from implicit a sign of understanding?

The trivial way of measuring an agent's capacity of understanding is asking questions about the context that is supposed to be comprehended. Figuring

out the interactions among the context components, the agent who answers the questions is said to have understood the context. That said, for the agent's comprehension, one thing is a prerequisite: background knowledge, the knowledge that is crucial for understanding the case or question. For this reason, background knowledge plus the context must be in the process of making inferences. Moreover, the agent can reorganize their knowledge when something is changed in the context or background knowledge. This change can be that there are black swans, or a single atom can be divided. Consequently, the capacity for understanding requires in- and off-context knowledge, the ability to de- and re-compose the components of context and background knowledge, and making inferences.

For instance, consider the following analogy. In a chess game, each player knows the pieces, the moves specific to each piece, and the object of the game. Each game, however, is different from the others since the movements can vary from game to game. Even a single different movement does affect and change the other movements. Thus, each movement creates a new composition/context up to which the players need to reorganize their strategies. Such a dynamic environment can also be shaped by the background information about the players, such as there might be a characteristic opening of a player so that the opponent can think of the chess compositions before the game. According to this analogy, an agent, who understands, behaves according to the changes in their environment. For instance, suppose that we are to explain artificial intelligence to a child and an adult. The words we use for each collocutor change; moreover, any contributions, comments, or objections of the interlocutors change the structure of the explanation. We, humans, then, construct new sentences whose components are chosen and organized in the course of communication. The idea is that, in the case of chess, there are pieces, each of which has its own rules. In communication, there are words, each of which has its own meaning and structural property, such as being an adverb or a conjunction. The players choose a piece and move it following the rules while having the strategies of winning circulating in mind; similarly, the interlocutors choose some words and utter them in a correct composition that reflects what the interlocutors want to mean. As

a result, understanding is a state of ability to re-combine at least two distinct components of a structure in the scope of a given context.

2.11.3 Machine-understandability

Suppose an agent is given a set of words containing "heavy rain," "river flood," "the top floors," and "two days continuance." Indeed, suppose that all these words are given in Turkish and whether or not the agent speaks Turkish is unknown. It is only when the agent can conclude -like "as an emergency, the residences of the ground floors may need to move to the top floors of the building, as the flood will continue two more days"- it is ensured the agent comprehends Turkish. In light of this, machine-understandability can be defined as the machine, as an agent, can manipulate the context-data within some broader contexts and make inferences in such an environment for the sake of a/some purpose(s). When the previous word set is given to the machine, it is supposed to reason and provide a result like the above.

It is always legitimate to define other kinds of machine-understandability; on the other hand, none of them has paved the way for realizing 4.0 versions of ISW. This bold claim leans on the nitty-gritty that realization of ISW 4.0, and thus machine-understandability, lies in the ability to manipulate data concerning their semantic properties according to a given context. Let us unpack this issue.

The machine is supposed to be an autonomous decision-maker that can interoperate different devices to realize Industry 4.0. We observed that there must be a unifying system that enables the machine to operate in an open world, viz., among several environments. Such a system is supposed to enable the machine to understand needs, demands, and situations, and then it can make decisions according to the changing needs, demands, and situations. For a scientific shift, the machine behaves as a colleague who can analyze previously generated scientific knowledge; observe *in vivo*, *in vitro*, and *in silico*, and generate hypotheses from the observations in light of the previous knowledge. In other words, the realization of Science 4.0 depends on the machine's ability to produce scientific knowledge that requires figuring out possible implicit relations among the phe-

nomena. Lastly, Web 4.0 is hoped to provide results where the users can see that their queries are categorized according to some contexts, and the machine can generate content. Namely, the machine is supposed to generate a categorization according to the query. The categorization process is conveyed according to the intended usage of the entities in the query and the sources. As a result, the users can see the websites listed for each category. Similarly, generating content from existing sources requires understating the content of the sources. Thus, all the 4.0 versions of ISW are looking for a machine that can manipulate data beyond the given properties of entities of a context and can reason in changing situations to discover, recommend, generate context, and decide in an open-world. Indeed, all these activities are not limited to well-defined contexts: they emerge whenever new evaluations over the contexts emerge.

Further, consider the following example. The machine detects the malfunction caused by a robotic arm on the production line and does the necessary things for troubleshooting, such as sending a message that specifies the situation to a technician or switch-off and -on the robotic arm. This is nothing but signalization. Nevertheless, in Industry 4.0, the machine is responsible for all the stages of production. That is to say, troubleshooting the robot is just a part of the work; indeed, the machine is responsible for deciding what is to be done at all the production stages during the downtime, for example, to not retard transportation. It may be that the machine delegates some work to other robots to save time or orders some pieces from another company to fasten the production. Hence, the machine *does* react upon and beyond the situation of a malfunctioning robot.

Under these circumstances, machine-understandability demands that all the possible states of entities be known, *in principle*. Why is that? The answer is twofold. Firstly, if the machine knows all the possible states of entities, it also knows their possible roles in different contexts. Concerning that, the machine can also find out the departures of meaning and roles of the entities in interacting with other entities. That results in the machine's ability to de- and re-compose context and background knowledge components. Accordingly, and secondly, knowing all the possible states of entities guarantees the establishment of background knowledge; since the machine can relate in- and off-context

knowledge when making inferences. However, how can the machine know all the possible states of everything? How can all the possible semantic properties of an entity and the possible interactions between entities be determined? It seems that the machine should have background knowledge in advance to understand the case or the question and then figure out the possible states of an entity. Besides, representing the background knowledge in advance sounds untenable for two reasons. Firstly, it means that the machine would have a monolithic structure, which makes background knowledge static. Secondly, there is nothing like representing background knowledge in an open world. A problem can be solved in various ways, where each way has its own structure and principles. Even the number of solutions is considered numerable; the ways to approach a problem are not easily predictable. This seems a taller order than the Semantic Web. We need to simplify our perspective towards what [machine-]understandability means.

Let aside all the previous examples of understanding, and let us put it in a simpler form: an agent understands what emerges when two states come together. Consider the following examples.

> $< fire|wood >$: fire has the property of burning among others, and wood had the property of being burned, again among its other properties. When these two entities interact, the mentioned properties will be in action, and *ash* will emerge.
>
> $< chair|sleeping >$: a chair has the property of having a flat surface, and sleeping has the property of lying on a surface that covers the body boundaries of what lies. According to these properties, a cat can sleep on a chair.
>
> $< water|hairdryer >$: water has the property of being an electrical conductor, and a hairdryer has the property of being powered by electricity. In the context of having a shower, these two properties reveal that using a hairdryer in the shower can be fatal.

All these examples show that, indeed, this is nothing but operating the semantic properties in a specific context. This is where machine-understandability rests. Thus, we can conclude that machine understanding is the ability to process semantic properties of data in a given context. Now, let us explain how the machine has been and should be processing semantic properties.

2.11.4 Processing Semantic Properties

Let us begin by providing little information about data processing. A programming language defines an object with a value and a data type. The data type states what operations are to be performed on the objects and to which type the value of the object changes with respect to operations. For instance, let us define x of string data type and set its value "laura." The machine can capitalize or reverse the letters of the string; concatenate two strings to form one string; returns the length of a string. All these are examples of operations the machine performs on the string data type. When we ask the machine to capitalize all the letters of a string, or simply to perform $capitalize(x)$, it returns "LAURA." Let us define y of integer data type and assign it "5." The machine can perform the predefined operations and functions on y along with other objects. For instance, when the "+" sign is defined as the mathematical operation of addition, it can operate on numeric data types. Thus, $y + 2$ gives 7. On the other hand, $x + y$ will error since adding a string to a number is nonsense. The machine cannot do mathematical operations on characters.[37]

That is, the machine can operate on data with respect to their values and data types. Hence, data processing means that the value(s) of object(s) change(s) under a specific data type.

The Semantic Web adds semantic properties as a new data component to become operable. A collection of such properties of entities is represented by annotations/tags specific to the context since Semantic Web's ontologies provide data

[37] Please exclude the languages, e.g., Python, where the addition sign (+) also operates as string concatenation.

organizations predefined. The inference rules for RDF, RDFS, and OWL are introduced to process these semantic properties. For instance, consider one of the previous examples where Laura being a woman was inferred from the schema and the assertion provided in the example, which is

```
:Mother rdfs:subClassOf :Woman.
:parentOf rdfs:domain :Mother.
:hasChild owl:equivalentProperty :parentOf.

:Laura :hasChild :Daphne
```

Now let us show how the machine processes the semantic properties according to the inference rules.

An OWL rule says that if `{?P, owl:equivalentProperty, ?Q . ?x, ?P, ?y}`, then `{?x, ?Q, ?y}`. Thus, from
`{:hasChild owl:equivalentProperty :parentOf .`
`:Laura :hasChild :Daphne}`, the machine infers that
`{:Laura : parentOf  :Daphne}`.

An RDFS rule says that if `{?P rdfs:domain, ?R . ?x, ?P, ?y}`,
then `{?x, rdf:type, ?R}`. Thus, from
`{:parentOf rdfs:domain :Mother . :Laura : parentOf :Daphne}`,
the machine infers that `{:Laura rdf:type  :Mother}`.

Another RDFS rule says that if `{?P rdfs:subClassOf ?Q .`
`?x rdf:type ?P}`, then `{?x rdf:type ?Q}`. Thus, the machine infers that
`{:Laura rdf:type :Woman}`, from
`{:Mother rdfs:subClassOf :Woman . : Laura rdf:type : Mother}`.

This is how the semantic properties of a data component are processed along with data type and data value.

In the case of machine-understandability, we claimed that the machine, as an agent, is to understand what emerges when two states come together in a context. Recall the example (Fromkin et al., 2018): "The assassin killed Thwacklehurst." Suppose that we ask the machine some questions about Thwacklehurst.

Only if the machine understands the statement can it answer the questions. As mentioned before, we can talk about machine-understandability whenever the machine can process the semantic properties. Thus, in this example, any question about Thwacklehurst can be answered from the semantic properties of an assassin and of killing, but primarily from an assassin. The semantic properties of an assassin -being a person, murdering someone important, planning, performing the act of killing in exchange for money or due to a fanatical adherence, and so on- give some facts about Thwacklehurst – human, important person, an opponent of something crucial; and the semantic properties of an assassin and killing together tell that Thwacklehurst is murdered, *ergo* dead. In this example, we aimed to show that the interactions between the semantic properties of an entity are crucial for understandability. Thus, machine-understandability hinges on processing semantic properties.[38]

Let us elaborate more on some of the previous examples. When "fire" and "wood" come together, ash emerges because the specific properties of fire and wood interact, resulting in ash. However, this interaction may be taken from another perspective, such as the need to get warm. The interaction between the property of realizing heat of a fire and the property of ignitability of wood emerges heat. Nonetheless, a caveat is in order. Straw also catches fire; it has ignitability properties, yet the straw fire lasts seconds and cannot be used for warming. Thus, the[39] essential properties of having ignition resistance and a low heat release rate of wood make it be used for warming. In the example of $< water|hairdryer >$, the specified properties were important in the context of having a shower and safety. On the other hand, the property of being an electrical conductor is not significant in the context of drying; instead, the property of being wet or making things wet (whichever is the preference) plays a role. Moreover, instead of being

[38] The relations between *context relevant* semantic properties guarantees the machine process. However, we will explain how this happens in the coming chapters.

[39] For the sake of simplicity, take these properties are the one that provides heating for more extended periods. Other related properties of wood play a role in heating, such as transferring heat or being composed of timber. Indeed, all these properties are in interaction, which we will mention in the following chapter in detail.

powered by electricity, the property of speeding the evaporation of the water of the hairdryer is significant for drying.[40]

Lastly, let us mention the intimate relationship between background knowledge and semantic properties. In his famous example,[41] Zambak gives three sequential statements as follows.

- A puts a book in a box.
- B takes the box and puts it in a car.
- C drives the car to Ankara.

According to this context, the place of the book can be asked. In a machine-readable system, the only information provided for the book's place is being in a box. On the other hand, in a machine-understandable system, the book is in Ankara because

- A puts a book in a box. The property of storing objects in a box and the property of being solid of a book say that the book is stored in the box.
- As B takes the box and puts it in a car, the property of storing the box and the common property of moving and taking means that the book is taken and put in the car along with the box. So, the book is now in the car.
- Since C drives the car to Ankara, the property of storing objects and of moving of car result that the book is in Ankara.

Lastly, suppose that there is another statement given:

- D takes the box and burns it.

The machine with a good sense of humor can say, "Which book?"

[40] Besides, iron can do the work in some circumstances since it also has the property of speeding water evaporation.

[41] These examples are taken from Zambak's lectures.

Therefore, the transformation of information given in the context is conveyed through the semantic properties of the entities, the properties that are crucial for the context. For instance, the property of having a color of a box or the property of having page numbers of a book has no interaction with other properties in this context. Hence, there are no inference rules in a machine-understandable system that processes the semantic properties; rather, the system itself operates on the semantic properties, and such operations themselves are inferences. The operations on the semantic properties held by the machine are to be asserted by semantic types; yet, let us discuss semantic types and how they function in the following chapters.

Consequently, machine-understandability means that the machine's ability to manipulate data concerning their properties that exhibit their meanings within a context. The machine-understandable data, Data 4.0, is the data that the machine can manipulate according to the context and process them with their semantic properties related to the context. This, however, means that entities are represented in the machine with all their semantic properties. According to context, the machine can choose the relevant semantic properties of entities, and the machine processes the relations between those selected properties. This new type of data is indispensable for realizing 4.0 versions of ISW.

Definition 5 *Machine Discovery: A collection of formally structured representations of phenomena through their semantic properties.*

According to this definition, Data 4.0 is data that the machine can manipulate meaning in- and off-context; namely, all possible contexts. That is why it is also called *machine discovery*: the machine can manipulate non-mechanically represented data; such processes end up with non-predetermined consequences.

2.12 Conclusion

In the first part of this chapter, we spoke of the new versions of Industry, Science, and Web when Big Data could cause a paradigm shift in these fields. Then, we

came to a point where such a shift requires machine autonomy. While discussing machine autonomy, we declared that the machine must understand and process Big Data. Accordingly, in the second part of this chapter, we claimed that none of the data definitions were made from a perspective that would lead to a machine-understanding system. Our difference was to make a classification by conceiving what data should be that the machine could discover without ignoring the existing data categories. Although we know that the data classification is *ad hoc*, we have also examined the available data categories. We also showed that we could not move to a machine-understood system without another category.

As it stands, the most straightforward data category would be Data 1.0, as we define it as anything that the machine processes. Thus, we have ensured that the focus is on what the data could be from the machine perspective rather than an empirical approach. In the second category, we put communication between machines at the center and defined Data 2.0 as the data category in which data flow between machines. The position of Data 3.0 was to point to machine-readable systems. Apart from the importance of human-machine communication in forming this data class, we have also seen that processing semantic data is obliged to human support and control. However, as explained in ISW 4.0s, all that is desired is for the machine to gain the feature of understanding. We argued that the machine would only be an active agent using Data 4.0. However, we evaluated neither understanding nor being an agent from a human perspective. Since what we understood from understanding was that existing concepts could be used with brand new combinations in new contexts. If defined through language, understanding brings words together in different contexts by following grammatical rules. A language speaker can utter an expression, putting together certain words in newly introduced contexts. The manifestation of this expression shows us that the person understands. So, we can say that the central issue of repositioning toward a machine-understandable data category is to automate how entities are handled in new contexts. In other words, the main reason for introducing Data 4.0 is to structure the existing phenomena in the machine; as such, ISW 4.0s can be realized. As a result of our work, we ended that Data 4.0 is a data category representing entities through their semantic properties.

However, there are four critical points to consider: How can the entities be theoretically and formally represented? How does the machine know which semantic properties to be processed? Assuming the machine can decide which properties to process, how can it operate on them? How can all of these be implemented? The following chapters deal with all these questions.

CHAPTER 3

DATA WITHIN ONTOLOGY

The previous chapter classified data into four categories and then dealt with the properties of Data 4.0, the only data category that can realize ISW 4.0s. The critical aspect of Data 4.0, the machine-understandable data, is that an entity is represented through its semantic properties. Representing entities in terms of their semantic properties has been introduced in the Semantic Web, yet it was just a part of the representation dependent on the context. However, we claim that entities are to be represented in terms of all their semantic properties, and the machine is to decide which of these to be chosen according to the context. When this is the case, it is required to find an ontological perspective that explains how properties define everything in the world.

Property is a fundamental category in philontologies. They are constituents of the intelligible aspect of everything; they even make themselves intelligible. For instance, a dress is green; green is a color; color is an attribute; an attribute is a category, and so on. So, we believe that the category of property must be studied philosophically; furthermore, an appropriate one should be utilized for this dissertation. Appendix Categories that Depict the World does the former. It analyses how ontological categories, particularly relations and properties, are studied in philosophy. Its purpose is to investigate the philontological characteristics of properties and other related categories. This chapter, on the other hand, studies the most suitable guiding ontological theory for a machine ontology found in light of appendix Categories that Depict the World. Then, this chapter offers utilization of trope theory, a philontological perspective for

representation in which entities are compresence of their properties. To put it more clearly, this chapter starts by explaining trope theory and why it is chosen for Data 4.0. Then, it will deal with computational aspects of utilizing trope theory, such as selecting properties according to a context and operating these properties. Lastly, it will introduce a new theory specific to machine discovery: the urtrope theory. We will see that the urtrope theory is machine ontology's philontological and computational foundation.

3.1 Phenomena into Data

3.1.1 Representing Everything with Tropes

Representation of entities with properties is not a novel idea. As a philontological stance, a trope theory defends that properties are the building blocks of reality; thus, everything is a particular combination of properties. In this part, we will provide an introductory level explanation of trope theories that suit the representational model of reality that we pose for machine-understandability.

The trope-only theories,[1] now on trope theories, state that tropes are the only ontological category that furnishes the world (Williams (1953), Maurin (2002), Campbell (1990)). More precisely, tropes are the single entity type, whereas other types of entities are accounted for in terms of tropes. Paul (2017, p. 33) states that "everything there is, including concrete objects like persons or stars, is a quality [property], a qualitative fusion, or a portion of the extended qualitative fusion that is the world-whole."[2] That is, concrete objects, abstract objects, spacetime, and relations; the world is constructed from properties. Hence, the one category out which the whole world can be constructed is properties, viz., tropes.

[1] There are other kinds of trope theories that acknowledge universals or substances. For details, see appendix Categories that Depict the World.

[2] Paul declares that she uses the term "properties" interchangeably with "qualities."

Anything –an object, an event, a property– is not an ordinary collection of tropes; rather, tropes are composed in a determined way to form entities. G. F. Stout (1940) defines *objects* as character complexes and states that "the complex unity of a character-complex is of a unique kind, differing from any other kind of complexity" (p. 126).[3] On the other hand, we do not know how to point out those relations. For this reason, although there are specific internal interactions of tropes for each entity, we can only talk about the emergence of the entity. The names given for the totality of the interactions vary; for instance, Paul utters "fusion" to indicate composition; Stout uses "concresce" in the meaning of interpenetration of tropes;[4] William speaks of "compresence."

We use the term "composition" to refer to the totality of the interactions of tropes, the interactions that emerge the entity. It may be helpful to make this concept concrete with some analogies. Firstly, let us take a tangram. A tangram consists of seven flat polygons, called tans, that are combined, without an overlap, to build various shapes. For instance, as in Figure 3.1, a combination of tans pictures a goose, and another combination pictures a cat. The tans have no meaning by themselves; they are just geometric entities; only their distinct combinations give rise to some meaning which are pictures of concrete entities. The colors of the tans have no significance other than making the units differentiable for humans.

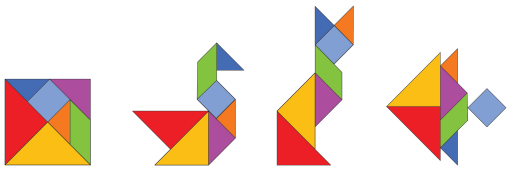


Figure 3.1: A set of tangs and some tangram figures

Another analogy is from chemistry. An *allotrope* is an element that can exist in different structural forms. These forms are different from each other not only in their forms but also in their physical and chemical properties. For instance, graphite and diamond are the allotropes of carbon. Moreover, the same

[3] Please note that his ontology is not trope-only.

[4] "Characters concresce in a concrete thing." G. F. Stout (1940, p. 128)

structural differences can also be observed in molecules called isomers, each of which has identical molecular formulas yet distinct combinations of atoms. Like allotropes, isomers can differ in their physical and chemical properties. These analogies help us describe how different compositions of the same units give rise to different entities.

We can further elaborate on the trope composition. From the previous examples, we can conclude that different trope compositions give rise to different entities. Moreover, an occurrence of a difference in a trope composition ends up with different facts about an entity. For instance, the cat figure in Figure 3.1 can be thought of as the cat standing and looking toward us. However, when the composition of tans changes, the cat can be thought of as lying on the ground, as seen in Figure 3.2.

Figure 3.2: Several tangram figures for a cat

These analogies suggest two things: (1) from basic units, complex things emerge, and (2) the way basic units compose characterizes the complex things. Tropes are simple in the sense that their compositions can construct everything that exists: an event -to rain- is a trope composition; a state -being happy- is a trope composition; Socrates -a particular-s is a trope composition; wet-trope, pleasure-trope, wise-trope are in these trope compositions, respectively. Consequently, trope theory can be chosen as the philosophical representational model of Data 4.0, where entities are represented by their semantic properties.

3.1.2 Trope Theory for Machine Discovery

Representing reality with properties is a well-established theory in philosophy, and taking tropes as the elements of being and representing the world with tropes and their compositions is philosophically grounded. Thus, we can legitimately abandon depicting the world with entity-relation dichotomy along with others. However, we still need a concrete explanation for this choice.

Our primary consideration is representing and processing entities in terms of their properties. We have arrived at this concern from the necessity that entities should be represented in a way that the machine can recognize them in all contexts. Otherwise, ISW 4.0s cannot be realized since all the data categories except Data 4.0 represent things in specific contexts. Further, a collection of semantic properties in a collection of contexts cannot portray the world in terms of properties since a brand new context may emerge new relations between semantic properties. In other words, a collection of representations of entities in terms of their properties only in specific contexts cannot be a solution to representing the world. For this reason, trope theory can help us represent everything in terms of its semantic properties.

The first reason for choosing trope theory is that, obviously, it provides philosophical foundations of phenomena into data. It is the closest philontological stance to Data 4.0. In addition to this, trope theory comes along with crucial virtues. First, tropes are compositional elements: entities are trope compositions, and they can be decomposed into tropes. A benefit of this virtue is that the machine can detect the possible contexts in which an entity can be. For instance, a person will be in the context of the plant consumer from the fact that they have a herbivore-trope, or in the context of politics from the social being-trope.[5] Secondly, vice versa is also possible: the machine can detect an entity in a possible context. For instance, the semantic properties of entities are pre-determined in the context of cloth washing. An Armani dress can be in this context because of its washable-trope, not because of its expensive-trope, or bungee-jumping cannot be in this context because there is no reasonable trope of it can be in the context. Consequently, in principle, trope theory allows the emergence of possible contexts of given entities and entities in contexts.

However, a virtue of trope theory contradicts our purposes: it represents entities in terms of properties, so it represents properties in terms of other properties. That causes an infinite regress, so this computational and philosophical

[5] How these contexts are determined will be explained soon.

issue must be addressed. Nevertheless, it suffices to say, for the time being, we can represent phenomena with properties and manipulate them on the machine thanks to trope theory. *Phenomena into data* is a revolution.

3.2 Selecting The Right Semantic Properties

Recall that a collection of semantic properties of an entity is represented by annotations specific to the context at hand in Data 3.0. To make this concrete, suppose we would like to create an application ontology of furniture. There is a furniture world that consists of everything about furniture, and the entities of this world are to be categorized. The classifications of ontologies start with a purpose, so firstly, we need to specify a task. The classifications of a furniture ontology built for an online selling site, that for production, and that for a warehouse, *must* vary. Each domain has its own constraints, related entities, relations, properties, etc. For example, a furniture ontology designed for an online selling site categorizes furniture according to its usage in a home, such as a study room furniture is grouped separately from the kitchen furniture. On the other hand, in a warehouse, furniture can be grouped according to its sizes: chairs, armchairs, tables, and so on. Thus, dining chairs would be in distinct categories in these ontologies. Consequently, the category labels draw the boundaries for entities so that Data 3.0 represents entities that gain definite roles or behaviors. Semantic properties of entities are set in advance; namely, they are predefined.

A chair can be used in the kitchen, study room, dining room, and garden; it can have different colors; it can denote some status; it can be made out of steel, wood, composite, and so on. As a Data 3.0, along with its fixed definition, "chair" has a collection of properties that shows up within a context that specifies the interaction of the other entities. To make this claim more concrete, please consider the following. "Chair" is "a seat typically having four legs and a back

for one person."[6] On the other hand, a chair can be used as a coffee table, bed, or step stool; a chair with casters can be further used as a serving cart. All these chair functions emerge from the interactions with other entities that are *barely* represented in a domain.

When the data is represented within a trope-based ontology, the perplexity begins by figuring out which roles of a chair come to the scene. On the other hand, the answer is quite apparent: the related properties pop up when the context is known. In other words, when the context is known, the roles and/or behaviors are also known. Indeed, representing the world in terms of entities and their interrelations is always context-dependent, as choosing the proper properties of entities is also always context-dependent. Thus, when the machine processes the semantic properties automatically, it must be context-sensitive.

Nevertheless, how can the machine become context-sensitive? Suppose we would like to represent things that can be done in the kitchen in the old fashion. When the task is washing the dishes, the list of entities does not include the coffee machine; if the task is making coffee, then the list of entities does not include a dirty pan. Hence, in this representation fashion, the entities and their interrelations are prioritized to specify the context. On the other hand, a dirty coffee mug may be an entity of making coffee-task; namely, the context should include other entities in some situations. When a change occurs in the context, the whole representation must be reconsidered; that means the context is static, thus the representation. At this rate, we find a direct way to prioritize the context; that is, the machine should recognize the context first. Then, any change in the context modifies the properties to be processed, not the entities. Moreover, there would not be another context to be represented; instead, there would be a dynamic context that fits a change.

Under these circumstances, we need to ask what determines a context. Previously we have mentioned domain ontologies and the Entity-Relationship (ER)

[6] Definition is taken from https://www.merriam-webster.com/dictionary/chair, on August 14th, 2021.

models as an elementary model similar to domain ontologies. Domain ontologies, designed for expressing meaning, are context-laden. That is to say, all the entities have specific senses/behaviors/roles within the context; furthermore, the entities determine the context. Suppose that two persons are traveling by the same train. They get off at the same station and walk through the same restaurant. One starts to work, and the other sits at a table and waits for the waitress. The passenger ontology of the former and the restaurant ontology of the latter categorize these two persons distinctly, for, in the former, they have the same roles. Yet, in the latter, they have totally distinct roles: the context changes, then the roles change. However, it seems there occurred a contradiction since we uttered at the beginning of the paragraph that entities determine the context. However, the roles are attributed to the entities; they are not considered by themselves. This toy example may seem redundant, but it significantly impacts the overall world representation. Secondly, an ER-model represents interrelated entities of a specific domain. A basic ER-model consists of entity-types that categorize the entities of interest and relationship-types that specify the interactions between the entities. An ER model is particularly useful for explaining the logical structure of databases, and each ER model is a domain standardization; they are not designed for expressing meaning. The fundamental difference between an ER-model and an application ontology is that the former focuses on data and the latter focuses on meaning. However, this does not change the fact that they prioritize entities to draw a domain's boundaries. Consequently, representations of data models or application ontologies mean specifying the roles/behaviors of entities that cannot lead to a dynamic context representation; thus, prioritizing the entities must be changed.

3.2.1 Relations determine a context

In order to realize machine ontology, so far, we have found that (1) everything – solid or abstract things, object or event, signal or textbook – must be represented in terms of semantic properties; the idea of which lies in trope theory, and (2) in order to process semantic properties the machine must be context-sensitive. Previous studies show that the machine cannot be context-sensitive when the

entities of a context are prioritized. We cannot prioritize semantic properties since they are found according to the context. However, we can survey from the notion of semantic properties. Consider some roles of a chair: its seat provides a surface for sleeping and providing a level surface. Since having a surface above the ground on which a human can step is a common property of both a chair and step stool; thus a chair can be used as a step stool. Alternatively, a chair functions as a bed for a cat since a chair has the property of having a surface whose area encloses a cat's body. Alternatively, the property of being solid makes a chair a doorstopper. Thus, we can figure out how a chair can function in a context thanks to its properties, yet what determines which properties are activated is the relations in the context. That is to say, it is not the chair and the counterpart entity that determine which semantic properties of them are activated; rather, the relations between them determine which semantic property of the chair is to be processed for the given context. Before explaining our solution to what determines a context, let us refer to two inspirational sources that shed light on our exploration of making the machine context-sensitive.

The first inspirational guidance is from philosophy. In his *Tractatus Logico-Philosophicus* (from now on TLP), Wittgenstein represents a contingent world in the sense that it is mutable, conditioned, and not necessary (Barroso, 2014). This dynamic worldview hinges on the idea that the world is everything that is the case (TLP: 1), which necessitates the *facts* being the basic structures of the world (TLP: 1.2). The crucial point here is that such a dynamic world is resolved in facts, not in things/objects (TLP: 1.1). An object can occur in various facts, in each of which the object gains its meaning. In other words, we cannot speak of the objects in themselves; their occurrence in a fact makes them knowable. The ontological and epistemological status of an object is based on a relation. Thus, this idea paves the way for talking about, for instance, using an instrument out of its functionality. For example, that a shoe can be used for opening a wine bottle is representable and thus meaningful in some contexts. Consequently, as TLP starts with the declaration of "the world is everything that is the case," the traditional categorical perspective that acknowledges entities as the building blocks of the ontologies is displaced. In other words, representing the world in

terms of entities and their interrelations is substituted with representing the world in terms of relations.

The second guiding inspiration is from theoretical physics. In his book *Helgoland*, Rovelli (2021) speaks of the world as a web of *interactions* and *relations* rather than objects. Everything in the universe – including electrons, humans, and planets – exists only in their interactions with one another (p. 59). For instance, think of a chair that is pink, has arms, and weighs 10kg. According to Rovelli, these properties do not emerge unless it interacts with something else. The chair is pink because it interacts with us, light, and alike. If there were an object without interaction, it would be devoid of all its properties: no interactions, no properties. Rovelli (2021, p. 62) states that "[t]he gist is that the properties of objects exist only in the moment of their interactions." He, then, continues "they can be real with respect to one object and not with respect to another," which means when no properties of an object are activated in some relation, the object is never there in that context. Consequently, the world is fundamentally made of relations rather than entities, or the entities are nothing but their interactions.

We want the machine to operate on the semantic properties to process meaning. To this end, we purported that representing entities with tropes is the most convenient ontological approach. Then, we claimed that figuring out which semantic properties of two entities are involved in a context is within the realm of machine-readability since the context is predefined; that is, the source and target entities and their interrelation are given. For machine-understandability, on the other hand, the context must be prioritized so that the semantic properties become automatically processable. Thus, it is not figuring out which semantic properties of 'human' and 'chair' should be processed when the relation is 'sitting;' it is rather figuring out which semantic properties should be processed when the relation is 'sitting.' Let us turn back to chair example, where the entities are chair, cat, human, and door, and 'chair' is the source entity, and the rest are the target entities. The relations between the source and the targets differ; thus, the relations determine which aspect of the chair is of concern. That is to say, a relation determines which semantic properties are *activated*. Con-

sequently, relations determine which semantic properties of source and target entities should be operated. This is the basic illustration of the relation-based approach, which must be extended to the context level.

Suppose that there is an unstructured science document. In order to machine understand it, all the entities[7] in the document are represented in machine understandable format, namely in Data 4.0. in accordance with the relation-based approach, the relations of the raw context must be listed so that the context is recognized through the relations. In such a setting, the meaning, namely the ontological statuses, of the source and target entities has no priority since their meaning will be determined after the context is determined. Here, it would be a mistake to determine their meaning by referring to the relation between them since the same relation may activate different semantic properties of the source and the target in different contexts. Thus, the relations of the raw context must be erected so that their combination would give meaning to the context. That is to say, relations determine the boundaries of a context, not the source and target. So, we purport that the machine can recognize a context through the relations, which determine which semantic properties of the source and target are activated based on the context. Consequently, relations tell us the boundaries of a context and which semantic properties of entities are activated in the context.

In sum, machine-understandability depends on processing semantic properties. Once the machine can operate on semantic properties, discovery, reasoning, recommendation, and content generation occur. To this end, we employed trope theory for representing data. On the other hand, we encountered the machine's problem recognizing which properties are to be operated. Data 3.0 consists of data representing entities with their annotated semantic properties that indicate specific roles or behaviors that depend on the context. On the other hand, Data 4.0 is the data representing entities via their semantic properties, which are operable by the machine. Thus, can the machine recognize which semantic

[7] Pay attention that all the literal objects are entities.

properties of the entities are in the process only when it recognizes the context. For this reason, context recognition must be prioritized. That is, the machine automatically finds the status of a context. We claim that the status of a context occurs when the combination of relations is figured out; thus, we seek a setting that realizes such occurrences.

3.3 Processing Properties/Tropes

In the previous parts, we slightly mentioned that types are crucial for processing. At first, we discussed that data types specify the behavior or the intended usage of objects, so a data type determines what operations are to be performed on the objects and constraints on the value transformations. Then, we mentioned that the meta-data, viz., annotations, determine the semantic types of objects; those types specify the behavior or intended usage of objects and semantic constraints on objects. In both cases, objects' values or semantic properties become computable thanks to typification. That is to say, a type assignment provides computational aspects to objects.

Following the same line of thought, we will purport that we need to typify properties; viz., tropes: Data 4.0 must have computable semantic parts; typification provides that semantic properties become computable. In the following, we will first investigate what a type is, then provide examples from logic, mathematics, and programming. Lastly, we will focus on utilizing types for processing Data 4.0.

Before moving on, a caveat is in order. Defining tropes, properties, and relations may not be very clear. Firstly, tropes are properties, and properties are tropes. They are the same thing for our machine ontological perspective. Indeed, "semantic properties" are the same thing. Relations, however, seem to be different from them. In appendix Categories that Depict the World, the difference between properties and relations is discussed at length, but here we will give only our machine ontological understanding of relations. We decided to utilize trope theory as the philontological foundation of machine ontology, according

to which there are tropes and everything is a composition of tropes. So, what are relations? Philosophically, relations are considered as more-than-one-place properties; that is, machine ontologically, the other way around, as we said earlier, properties are unary relations. As the whole machine ontology story relies on the relation-based approach, we conclude that relations are either tropes or trope compositions. We will return to this matter while discussing their formal descriptions.

3.3.1 Typification and Type Theory

In philosophical parlance, type has several senses, such as universal, a species of universal, a law, or the sets of its tokens (Wetzel, 2018). Putting aside its metaphysical connotations, types are beyond classes and categories since they define not only the specific characteristics of terms but also their allowable interactions. By the allowable interactions, we mean that types highlight the states of affairs or say possible facts about the terms/objects. For instance, restaurant types specify what kind of dishes to be served along with what to wear, or hair types specify what kind of hair conditioner is to be chosen. In a nutshell, types are useful for the classification of entities along with specifying constraints on them.

Russell first introduced the idea of type to eliminate paradoxes in the set theory, such as the set of all sets cannot and must be an element of itself. Russell's type theory distinguishes properties into a hierarchy of types so self-predication can never happen. For instance, the types of numbers are differentiated by the types of sets of numbers, the types of functions from numbers to the sets of numbers, and so on (Benzmüller & Andrews, 2019). Thanks to such distinctions, paradoxes in the set theory are abolished. The initial purpose of the type theories, then, was to define mathematical objects with the previously defined objects in order not to commit impredicative definitions, viz., self-referencing definitions. Thus, type theories in logic and mathematics aim at formalizing terms and operations on them as well as formalizing types for each term.

The idea of types was utilized outside of these fields, especially in programming languages. In programming languages, every term has a type, which defines the meaning of the term and the operations that can be applied to it. For instance, when a variable, say "3", is defined as an integer type, then arithmetic can be applied to it. However, it has no further mathematical features when it is defined as a character type. Thus, a type in a type system defines the expected properties of the type and operations on the objects.

In a type system, the definitions of the expected operations on objects are of the most importance. Such a system executes constraints on object interaction; in other words, a type system allows only consistent interactions between objects (Cardelli & Wegner, 1985). Thus, type systems enable that developing a program and verifying its correctness can be proceeded in a single system so that the quality of the systems by optimizing memory efficiency and enabling the detection of errors before they become run-time problems improves. For instance, in the C programming language, an integer is of one of the following types: integer, long integer, long-long integer, short integer, or unsigned integer. These types save memory usage. Or, in Java, the compiler returns *incompatible type* error when a value is forced to be stored into an array of a mismatched type. To summarize, a type system introduces types and their terms and formalizes reasoning with such typed variables.

In ontologies, types are defined like universals, as the things that classify. ? (?, p. 535) says that "Aristotle" is classified and described by types of "human" and "philosopher." As types define the classes of entities, the semantic properties that entities carry are in the form of types. More precisely, in Ontology 3.0, tags typify entities and relations, and those tags are operated according to the inference rules. On the other hand, typed properties are processed concerning a specific domain. For example, "Aristotle" is of philosopher-type, which is of human-type. In another context, he can be typed as "male" and "founder of the Lyceum." Or consider the following. We cannot know the ontological statuses of 011, 11, and 0011. We need to know the context in advance; namely, we need to know their interactions. For instance "11+11= 22," "11+11=110", and "011=11=0011" are of different contexts. Indeed this is not even enough:

"11+11=22" and "11+11=22" may be of different contexts; think that the former calculation is conveyed on the octal system and the letter is in the decimal system. That is, a type defines the allowable interactions; each typification defines entities' context-based roles/behaviors.

Consequently, types are used to draw boundaries between entities; such as in the realm of arithmetic, there are three types of integers; zero, negative, and positive. These kinds of boundaries offer a convenient formalism to entities and their implementation. Thus, type systems impose constraints that guarantee consistency of the systems. As in the case of Ontology 3.0, tagging is a kind of a type system that specifies the semantic properties of entities and the operations on entities in a context.

Let us recap some critical arguments of this work. We argued that everything in the world gains meaning within an interaction. An entity by itself alone is unknowable, thus undefinable. In other words, an entity gains its ontological status with relations that relate it to other entities. For this reason, it will be futile to typify the entities for our aim of machine-understandability, although typifying objects has worked for several tasks, such as machine-readability. Because annotations typify entities, the metadata is operated through the inference rules, all of which are specific to a domain. This is valid for both data types in programming languages and machine-readable systems: the entities are typified for some expected behaviors/roles. Nevertheless, for machine-understandability, the machine is supposed to assign types automatically; yet, this has to be done through semantic properties that are determined by the relations. In this respect, the entities cannot be typified, but the tropes and their compositions, because only then can the operations on semantic properties be specified. It bears repeating that the entities are trope compositions; viz., an entity is represented by its actual and potential semantic properties, or tropes. Thus, not the entity highlights which semantic properties to be processed, but the relations determine which semantic properties to be processed. In short, machine-understandability postulates the typification of tropes and trope compositions. On the way to machine-understandability, first, tropes are to be typified; second, the rules for operating tropes and trope compositions must be specified; and lastly, a com-

putational model for operating the rules must be promoted. Let us explore the typification needed for machine discovery.

3.3.2 Typification of Relations

Typification is at the center of this work because representation is not enough alone; processing the representations is required. In this part, we will investigate relation types and typifying relations. We have chosen relations types since everything else gains meaning through relations.

In a machine-readable system, all the entities are supposed to be typified by human power; or in logic or programming, every variable should have a type assigned by logicians or programmers. It seems quite impossible to typify all of the relations for all domains.

We argued that trope theoretical representation of reality *does* pave the way for machine-understandability. As everything can be represented with tropes, so do relations. At this rate, it is legitimate to claim that *some* trope compositions must be typified. We uttered "some" since some trope compositions are relations, some trope compositions are events, and some are numbers. Thus, the focus is on the trope compositions that emerge relations. In other words, tropes are the elements that determine the relation types.

We are speaking of a type system that specifies interactions between semantic properties and the operations on those semantic properties in a context, and relation types are assigned to trope compositions to distinguish between relations. For instance, the relation type of locomotion is differentiated from the relation type of motion with respect to their trope compositions.[8] As the type theo-

[8] One may object that locomotion is not a relation, yet it is spoken of as a relation type. For the time being, please do not let nominalization perplex you, and please beg this usage. We will see that intensional logics are of importance in our work. The principle of substitutivity of equivalent formulas fails in intensional logic; thus, self-predicable and non-self-predicable properties seem to not co-exist in a system (Bealer & Mönnich, 2003). For instance, "being an abstract is abstract" can be presumably a valid statement, yet, "being gold is gold" cannot. Indeed, being-gold is a trope, and gold is an object, that is, a trope composition; the trope composition of gold contains a being-gold trope. On the other hand, gold can be a trope that is of color type. So, being a complex or primitive relation hinges on the type of trope or the trope composition. Bealer and Mönnich (2003) solve this

ries offer, typifications provide relation hierarchies; in other words, the relation types subsume their subrelations. Yet, a relation type is a trope composition, so as its subrelation. Therefore, there must be partly structural identity between types and their subrelations; such a structural identity must exhibit itself in the trope composition at hand. This reminds Simons (1994)'s work, "Particulars in particular clothing: Three trope theories of substance," where he deals with the problem of particulars and universals and develops his *Nucleus Theory* on the combination of aspects and thus advantages of both bundle theory and substratum theory. Leaving aside all the metaphysical debates, let us concentrate on how he constructs a collection of tropes as an individual. Simons suggests two constituents of a particular. The first one is called *nucleus*, which is "a collection of tropes which must all co-occur as individuals" (Simons, 1994, p. 567). In other words, a nucleus is made up of mutually dependent definite tropes. This is the foundational system of a particular. The second one is *supplementation* by tropes, a collection of tropes of *certain determinable kinds*, whose tropes rely specifically on the tropes in the nucleus. This is the contingent system of a particular. The crucial difference between the nucleus and its complementary is that every trope in the nucleus is foundationally related to every other trope in the nucleus, and none of the tropes is foundationally related to any other trope that is not in the nucleus; and the contingent part, whose tropes depend on other tropes in the nucleus specifically, is connected to the nucleus.

To elucidate this crucial difference, ponder on the following. The tropes in the nucleus necessarily depend on each other and do not admit a trope to come in or go out. The tropes in the supplement are dependent specifically on the nucleus, namely, the tropes in the nucleus. That is to say, as the supplementary consists of tropes of certain determinable kinds, we can speak of a certain kind of tropes, but not an individualized trope in that kind, so some tropes in the

perplexity by showing that intensional logic is committed to the existence of intensional entities and that "intensionality is a necessary feature of a type-free system that allows for unlimited self-reference in the form of an unrestricted abstraction principle" with a firm claim that failure of substitutivity is the best criterion for characterizing intensional logic (p. 239). Unfortunately, tropes are extensional entities. Nevertheless, in a formal system for type-free property theory, properties can occur in both subject and predicate position (Orilia, 2000). Thus, a trope can be a subject and predicate in a type-free property theory, and its valid self-reference is guaranteed.

supplement can be replaced, yet the particular preserves its identity.[9] Hence, the nucleus acts as the core to the contingent tropes, and accounts for their all being together (Simons, 1994, p. 567).

Simons asserts several advantages of his Nuclear Theory, which is said to be flexible. The first source of flexibility is about nuclei, which can be different sizes and complexities. There can be particulars without a supplement; they are all nuclei. Or, there can be a substantial collection of tropes without a nucleus. In this case, an individual trope is associated with some particular tropes in the course of its life, and each association is necessarily contingent. This means that such a trope cannot exist alone and can be associated with different tropes in its lifetime. Thus, this is not a case for free single tropes but a substantial collection of tropes. Lastly, there can be a single nuclear trope that serves as a genuine substratum to its supplement. Recall that the supplement system is a collection of certain determinable kinds of tropes. So, a trope in the supplement can be replaced by another trope from the same determinable kind; however, the nuclear trope cannot be replaced: its annihilation dispels the supplement.

The second source of flexibility is about the supplement. A supplement may consist of clumps, each having its own subnucleus. In other words, some properties are complex, which have their own nuclei. Moreover, there may be clumps associated with each other in the contingent system. Thus, there can be tropes with nuclei either in the foundational or contingent system of a particular (Simons, 1994, p. 569).

This vein can be used for elucidating the typification of relations. Like any other entity in the trope theory, we claim that a relation type consists of two parts: core and peripheral. The core is the foundational trope composition that ever changes, and the peripheral is the contingent trope composition; the different configurations yield the subrelations. Thus, composition differences in the core and the peripheral specify the subrelations, and all subrelations have identical

[9] Simons (1994) also solves the problem with Bradley's regress by introducing contingency in a part of trope compositions. For details see Maurin (2018).

tropes in the core of the relation-type. For instance, there is a locomotion-type, two of whose subrelations are 'walk' and 'trot.' The trope composition of 'walk 'and the trope composition of trot have in common the tropes in the core of the trope composition of locomotion, and they necessarily have different trope compositions from locomotion. It may be a brisk-trope that is not at the core of 'walk,' but 'trot.' A caveat is needed here: the core of locomotion is not the same as the core of its subrelations. Each subrelation has its own core and peripheral.

Now that we understand the relationship between a relation-type and its subrelations, we can move on to elucidating how relation-types work and how this type theory contributes to computation. Without further ado, a relation-type specifies (1) operable semantic properties (thus, which entities *can* interact), (2) operations on those properties, and (3) their constraints. (1) Recall that an entity cannot give its types by itself. A water bottle and a hand-sized rock do not share the same typification in the context of containing liquid. On the other hand, both share the same typification in the context of being a doorstopper, as both of their trope composition includes solid-trope, for instance. That is a relation-type highlights which semantic properties are to be operated. Thus, each typification is of a contextual usage, defining the operable semantic properties, not the entities. Suppose a seeing-type activates having-capacity-of-seeing-trope and reflecting-light-trope; rather than an eye and a diamond. Moreover, (2) the allowable interactions between semantic properties are determined by the trope composition of the relation-type or one of its subrelation. When a relation-type is decomposed into its tropes, the machine can figure out the allowable interactions between semantic properties and what emerges after the interactions. Additionally, invoke-type of cause and occasion-type of cause have different trope compositions, so they activate distinct semantic properties. Lastly, (3) the trope composition of a relation-type specifies the constraints on operations. Consider the following statements constructed with the same relation: "Laura lives in Ankara" and "Lemon lives in Ankara." The trope composition of the relation-type of 'live' determines whether or not these two statements are valid, such that the relation-type allows a transition from a trope composition

of Laura and a transition from a trope composition of lemon. Here are the alternatives: the relation-type of 'live' in these two statements allows a transition between Laura/lemon and Ankara for either, or neither, or both of the statements. Consequently, typification of relations paves the way for constructing type structures, which are computable.

How about determining a context concerning the relations. The relation types determine the computable aspect of a context. That is to say, a relation can be of different types; however, the relational compositions can occur only among specific types. Recall one upmost important aspect of machine ontology: it is a relation-based approach. Thus, relations have types, and the operations between relations are defined. So, a context is nothing but a web of operable relations. For instance, the relation type-A can occur only if there is the relation type-B. If a context has two relations a and b, and it is known that a is of type-A, then b should be of type-B. The types of the rest of the relations are determined in rapid succession. Eventually, the context is determined when all relations gain their types.

3.3.3 The Need for Types of Types

In the scope of this work, a type theory is crucial for computing semantic properties. Relation-types set boundaries between relations, specify operations with relations and put some constraints on these operations. Nevertheless, this is just the beginning: for machine-understandability, the machine can typify the types repeatedly when necessary. Think of machine-readable systems, where metadata contains types of entities, and all the semantic operations are held through those types. The machine cannot go one step further since there must be another collection of metadata about the metadata that may need to be specific to the context. For instance, consider that we are given two statements: "Socrates is a philosopher" and "All humans are mortal." If the machine were to infer "Socrates is mortal," it would have annotated 'human' to 'philosopher'

precisely for this context.[10] Thus, the machine must be able to interpret types specific to the context, so that it can be generative.[11]

In the machine-understandable systems, thus, the machine can interpret different types by decomposition. To make this more concrete, please consider the following examples. A location-type has several subrelations, such as stay-, live-in-, dwell-, and settle-type. O the other hand, 'stay,' as a location-type relation, has its own type, so there are subrelations of 'stay;' 'romantic-stay;' 'compulsory-stay;' 'home-stay,' to name some. Indeed, there can be several kinds of 'compulsory-stay,' such as 'custody-stay' and 'intensive-care-stay.' These relation types occur in different contexts and activate different semantic properties. Furthermore, 'location' can be of a type; for instance, it has a supertype: is-in-type.[12] As subtypes depend on supertypes (recall the nucleus theory), operations are performed on subtypes when necessary and on supertypes when necessary. The machine can decide which one is on the scene by figuring out the context from the web of relations, each of which is on the web due to their types. Thus, it is legitimate to consider the following interwoven types. Exemplification-type has 'illustrate,' 'explain,' and 'describe' as subrelations; and 'exemplification,' by itself, is a subrelation of is-a-type, along with 'subsumption' and 'specification.' And, there are genus-subsumption and determinable-subsumption as subtypes of 'subsumption.' Yet, there are more: 'subsumption' is a subtype of both 'is-a' and 'Brook's architecture.' For instance, 'subsumption' and 'exemplification' can be contrasted and compared via their tropic compositions, and similarities and differences between is-a-type and Brook's-architecture-type of subsumption can be elucidated by decomposing the trope compositions.

All these investigations have brought us to an essential turning point: the machine needs to manipulate types. In other words, on the way to realizing ISW 4.0s, we need to find a way that the machine can process types, types of types,

[10] Another possible annotation of 'philosopher' can be 'profession.'

[11] However, we have already mentioned that entity-based annotations cannot pave the way for a generative system. The machine requires annotations inserted by humans for each context; thus, neither can it generate types of types nor make inference intercontexts.

[12] Another subrelation of is-in-type is 'contain.'

and types of types of types. Thus, once the machine can process types of types, it can also process types of types of types, and so on. Moreover, it can also decide whether sub- or supertype to be processed. It is worth reminding that the machine's ability to typify the typifications provides a generative system by enabling interpreting types from various dimensions; it gives a computational aspect to any trope composition; consequently, the machine discovers explicit interactions.

On the other hand, the mentioned type theories offer a hierarchy of types that enlarges properties *ad infinitum*; as such, the objects of each type form a new domain of the next type. For instance, the objects are of type 0; the first order properties are of type 1; the totality of first-order properties is the second-order properties that are of type 2; and the totality of second-order properties is the third order properties that are of type 3; and so on. In such a hierarchy, there is quantification over relations and relations of relations; but over types. Yet, machine-understandability requires not only regimenting relations into types but also types into types. Thus, we need a type theory to define and compute types over types. Fortunately, there is such groundwork for speaking of types of types to which we can refer.

3.3.3.1 Expressing Types of Types

The *Entscheidungsproblem* was one of the problems in Hilbert's Program that surveys whether there is an algorithm that can decide the truth or falsity of any statement. While dealing with functions as rules in his λ-calculus, Church introduced his definition of effectively calculable, *Church's thesis*, and showed that the *Entscheidungsproblem* has no λ-definable solution. Besides, he encountered the inconsistency of $\lambda\ x.xx$: a predicate can be applied to itself. Church borrowed Russell's solution to solve a similar inconsistency: terms are to be typified. Then, the simply typed λ-calculus was developed.

The part that concerns us begins now. Curry discovered that there is a correspondence between types of functions and propositions. That is, every proposition in propositional logic can be read as a type of a function in the simply typed

λ-calculus. On the other hand, Howard extended this correspondence to predicate logic and showed a correspondence between the simply typed λ-calculus and natural deduction (Wadler, 2015). That means representing and computations on the predicates of predicate logic can correspond to formations and operations on the dependent types. The combination of these two results is known as the *Curry–Howard Correspondence*, or the Curry–Howard isomorphism, which identifies propositions as types.[13] For instance, the predicate $Even(x)$ is the type of proofs that "x is even."

The Curry-Howard correspondence has a great repercussion among the constructivists, who assert that to prove the existence of an [mathematical] object is to construct that object explicitly (Çevik, 2019).[14] In other words, asserting the existence of an object is just constructing that object. For instance, a method for finding such a number must be provided to say that there exists a number with a property. A repercussion in logic of the correspondence among constructivists is *intuitionistic type theory* or *Martin-Löf's constructive type theory*, or simply *Martin-Löf type theory*. In Dybjer and Palmgren (2020)'s wording, "*a proposition is the type of its proofs* is fundamental to intuitionistic type theory."

According to Martin-Löf, so to intuitionistic type theory, all the logical connectives and quantifies are instructions for constructing a proof of a statement that includes the logical expressions; as such, the identification of propositions and types follows that logical constants as type formers, predicates as dependent types, and relations are families (Dybjer & Palmgren, 2020).[15] Moreover, he defines a type-theoretic universe U, which is closed under all type forming operations. On the other hand, as U is itself a type, it cannot contain itself; otherwise, a paradox familiar to Russell's would emerge. Then, he expands this idea to a countable hierarchy of universes: A universe, say U_0, is a type that

[13] The one-to-one correspondence between propositions and types preserves between proofs and programs, and simplifying proofs as evaluation of programs (Wadler, 2015). This correspondence has tremendous results in proof theory, semantics, and logic, to name some; however, the limits of this work cannot allow us to elaborate further on this observation.

[14] As such, they accept neither excluded middle nor double negation.

[15] An exploration of Martin-Löf's type theory is beyond the limits of this work. The crucial point is to grasp that there is a formal system that includes the typification of types.

includes all the types, but it is not contained in itself; the larger universe, say U_1, is a type which contains all the types of U_0 and the type U_0; similarly, the larger universe, say U_2, is a type which contains all the types of U_1 and the type U_1, and so on. Thus, this computable hierarchy of universes, U_0: U_1: U_2: U_3: ...,[16] is nothing but a hierarchy of types. In this way, each type has a type; namely, typification of types is possible (Dybjer & Palmgren, 2020).

There are two takeaways from Martin-Löf's type theory. (1) New types can be defined without committing paradoxes, and (2) definitions of the types and computation in types, but also in types of types, became possible. Consequently, Martin-Löf's type theory can be the formal basis for relation typification in the trope-ontological approach for machine-understanding. Any valid trope composition can be expressed as a type, and according to its compositions, there can be super- or subtypes of the type in question. Defining and computing the types hinge on the trope compositions. Although the compositions are not formally defined yet, Martin-Löf's type theory provides axioms and rules of inference for types. Hence, Martin-Löf's type theory is a productive type system in which any trope composition can be expressed as a type, and new types can be defined whenever necessary.

3.4 Tropes as Types

In the last part, the puzzlement of typification of types is solved. On the other hand, the mentioned type theories are known for admitting the sharp distinction between objects and properties. However, this approach is not valid in a machine ontology, as everything is either a trope or a composition of tropes. In this case, types must be attained to tropes and trope compositions. Indeed, in a machine-understandable system, the relations must be typed according to their trope compositions. However, no property is not a relation: to typify relations, in the most profound sense, is to typify tropes. Namely, the relation-based approach is

[16] $a : X$ means that a is of type X.

nothing but a trope theoretical approach. Now let us investigate tropes as types more.

A trope can be of different types, so as a trope composition. For instance, a 23-trope has different types in different settings: as a character for Michael Jordan's jersey number; as an integer for the total number of candies Laura ate today. A green-trope has many types as well. For instance, it is of color-type, fresh-type, or environmental-friendly-type. 'Student' as a trope composition also has many types: a human; an occupation. Even individuals, such as Socrates, exemplify many types: have-ugly-nose; put-to-death. That is, each typification occurs in a context that has its own ontological structure. Put differently, a trope or a trope composition behaves differently in different contexts. However, we know that the relation types specify different behaviors of tropes or trope compositions, and we can say that trope types specify different behaviors of tropes. Moreover, considering that as in a type theory, all properties are *typed*, so are tropes, it is legitimate to generalize the relation-based approach to the trope theoretical approach. That is, relation types are based on trope types intrinsically.

The gist of the above discussion is here: relations specify a context. Their occurrence in a context must be *a priori* organized. The otherwise cannot be reasonable since the otherwise means that there is no context at all but a bunch of representations. Thus, relations are in a context with specific roles/behaviors, and typing rules specify those roles/behaviors. The machine then detects the relations, figures out their types, and specifies the context. Once the context is specified, the semantic properties of the source and the target entities are activated. Nevertheless, relations must be considered on two grounds, which are often interweaving. At the first ground, it is an entity in a context; the standard annotation is seen in Ontology 3.0 and Web 3.0. in a statement, for instance, a relation is an entity between a source and a target entity. The machine specifies a context with the help of relations at this ground. On the second ground, however, relations are tropes. This is the machine ontological status of relations. Thus, when a type of a relation is spoken of, we must be aware of which ground of relation is considered. A relation is of the first ground when the contextual level of examination is surveyed; a relation is of

the second ground when the entity level of examination is surveyed. However, determining the ground of the relation is not always easy and even unnecessary: the relation types can be reduced to trope types, and tropes are the building blocks of everything. Consequently, trope types are the computational units. Types of types are again trope types, as types of types of types. For the sake of simplicity, when we say a relation type, it refers to a trope type that operates at the contextual level.

3.4.1 Extensionality of Tropes

Super- and subtypes of relations can be easily detected due to their trope compositions, where the composition differences in the peripheral specify the subtypes, and all subrelations have the identical core of the supertype. We have seen that the relation types can be reduced to trope types, and there are types of types of tropes as there are types of types of relations. However, this results in puzzlement: can there be types of types of tropes, the building blocks of reality that cannot have any part?

Ponder on the following cases. Firstly, think of a yellow-trope that can only, for the sake of simplicity, be of either warm color-type or primary color-type. How can the machine determine the type of the trope within the trope compositions? Secondly, assume that triangular-trope and trilateral-trope behave the same in all circumstances. Then, should the machine take triangular-trope and trilateral-trope to be identical?[17] And next, recall that a green-trope can be of color-type, fresh-type, or environmental-friendly-type. Indeed, a green-trope is of subtypes of color-types, such as a cool color and, at the same time, a secondary color. How can the machine determine super- and subtypes of tropes? Lastly, without any information on the type of a 23-trope, what does the machine give as a result for "23+23" if it can calculate it?

[17] In the scope of this work, please, ignore the metaphysical analysis of the *hyperintensional identity of properties*. Besides, we will mention intentional theories in the following parts.

That tropes are typed makes them extensional (Cf. Moltmann, 2013). So, tropes of the same type behave the same; then, two tropes must be the same when they interact with the same tropes in the same way in all cores of the trope compositions in the same way within the constraints that the type specifies. For instance, a yellow-trope of a warm color-type always interacts with the same tropes in the same way in all cores; and a yellow-trope of a primary color-type always interacts with the same tropes in the same way in all cores, where there must be at least one different trope or interaction in the former and the latter. Thus, it may be the case that the trope(s) that interact(s) with and/or the interaction(s) between tropes in yellow-trope determine(s) the type of it. In this case, the tropes that specify the type mutually depend on each other in all cases. For instance, the trope(s) that specify/specifies a yellow-trope as a type of primary color interact(s) with yellow-trope in the same way in all trope compositions. This is untenable since what determines a yellow-trope interacts with should be its type; not (an)other trope(s) that interact(s) with a yellow-trope determine(s) the type of a yellow-trope. That is, the types are computationally prior to trope interactions. Similarly, a triangular-trope and a trilateral-trope have the same extension. On the other hand, a having-side-trope should have a different kind of interaction than that of a having-angle-trope. Because the spotlight is on the feature of having-side for the former. Bealer and Mönnich (2003) discuss this issue within the intentional entities approach, where for any object x, the proposition that x is triangular is a different proposition from that x is trilateral, even if it is absolutely true that anything that is triangular in any world is trilateral in that same world. Just like the sense of Morning Star is different from the sense of Evening Star, the senses of the property of being trilateral and the property of being triangular. Thus, saying "it is trivial that an equilateral triangle is an equilateral triangle" is not that trivial (Fitting, 2020). Next, there are several results for "23+23" according to the elements' types. If either of the 23s is of character-type, and "+" only operates on number types, the result is a type error. If "+" operates on character-type as a concatenation operator, then the result is 2323, when both 23s are of character-type. If "+" operates on number-types, then the result is 46 when both 23s are of integer-type. In all of these examples, the types need to be known in advance to reach

the result. However, for machine-understandability, the machine is supposed to assign types to tropes so that it can operate them with respect to the context.

Lastly, in the example of a green-trope, we saw that there are sub- and super-types of tropes. As tropes are typed, again, the machine cannot decide which type the trope is unless it is annotated manually. Of course, typing rules provide a list of types of types of tropes; however, we are discussing the 'primitive' trope types, primitive in the sense that such tropes are not a composition of any other tropes. Therefore, such trope types, which typify primitive tropes, are just U_0 in Martin-Löf's theory, which is given *in advance*.

The type constructors and type assignment axioms can be specialized for the case of tropes, as it has been practiced for many years in logic and computer science. Hence, a system in which tropes are automatically identified with types is constructed, then a type assignment system occurs that includes types of tropes. The idea that tropes as types are theoretically conceivable thanks to intuitionistic type theories; however, the very same theories are notoriously known for that the type-theoretical hierarchy multiplies tropes *ad infinitum*. For instance, a green-trope is of environmental-friendly-type; an environmental-friendly-trope can be of construction-style-type; a construction-style-trope can be of cultural-identity-type; and so on. However, the realization of machine-understandability hinges on the machine's ability to assign types automatically. Thus, the fact that U_0 is given in advance is not the primary issue here; it is rather if the machine cannot decide the types automatically because the 'primitive' tropes got their roles in advance. It means that we have not taken a step forward from Ontology 3.0. Hence, the tropes must be *untyped* so that the machine can automatically decide the 'primary' trope types. It needs to be untyped; it is only when the machine can automatically assign a role to it. However, this is contradictory as the tropes are extensional from their very nature.

3.4.2 Self-reference of Tropes

Consider a well-known self-reference example: the property of being a property is itself a property, or the property of being abstract is itself abstract that is a

property, presumably. Besides, recall that type theories forbid self-reference in the very first place for eliminating paradoxes. So, what is the case of types of tropes?

Self-reference is inevitable in typification. Being a property is a property, or presumably, being abstract is abstract. One may claim that being abstract-trope is not the same as abstract-trope as the former is a relation –a trope composition- and the latter is a property –can be a *primitive* trope.[18] However, this is not the prerequisite that tropes must be self-reflexive. The gist of this issue is the fact that self-reference is a crucial feature for machine-understandability. In a self-referential system, each unit in the whole has its identity so that each part in the whole can be differentiated from another. Then, self-reference can construct the basis for a complexity index, which is crucial for figuring out deeper complex structures and processes in the whole of the system (Hempel, Pineda, & Smith, 2011). Thus, the representation system of the machine should be self-referential so that interactions between entities in any context can be designed and described by the machine. On the other hand, tropes are the building blocks of the representation system of the machine, and they dwell in a type theoretical realm where self-reference is forbidden. A further issue concerning this feature of tropes is their building blocks of reality. If the building blocks of reality were typed, for each context, we would need to specify the content of the building blocks. This is nonsense since this idea is against the meaning of the concept of "building block." A building block/a unit, an atom (in Democritus' sense), is used for designating things that do not behave differently in a different space-time. Only its compositions give rise to something that can be named. Hence, the extensional feature of tropes is, again, untenable from a machine-understandability perspective.

In conclusion, tropes, as they are typed, are in a context with their specific behaviors/roles. Each role/behavior solely represents one aspect of a trope. Representations, on the other hand, should encompass all possible states, which are

[18] Of course, being yellow is not yellow.

determined by the possible interactions in a context. Moreover, not only potential interactions of tropes but also self-reference cannot be represented in a type theory where tropes are typed. Consequently, the proposed trope and type theoretical approach is necessary but not sufficient for machine-understandability. Tropes are extensional, and any explanation about their whatness goes *ad infinitum*. If tropes were not extensional, they could be self-referent and atomic. So, we must invent something new for Data 4.0: non-extensional, viz., untyped ontological things.

3.5 The Urtrope Theory

We have seen in the above study that type theory is essential for making semantic properties processable, and further, automatically finding the types of the types is essential for machine-understandability. This section lays the foundations for constructing an ontological type theory that will lead to generating types of types. At this rate, first, how the properties of an untyped system should be will be examined, then a theory will be presented while preserving the trope-theoretic ontological structure.

3.5.1 Untyped Formal Systems

A solution for the mentioned issues arises when the tropes are stripped from their types; however, this is untenable since the aim of employment of a type theory is to assign types to tropes to make them computational units. Thus, tropes are *dressed* representations, in the parlance of Cardelli and Wegner (1985), and such dresses determine which tropes are to interact with each other: a type preserves the underlying representation. In the following, we will investigate type-free property theories that allow *naked* representations and then lay out how we should construct an untyped system for machine-understandability.

Type-free property theories are introduced in order that properties can be self-predicated. (Orilia, 1991). There are two interrelated, but can be separately suggested, purposes for introducing a type-free property theory. The first one

is developed in order to circumvent Russell's paradox. This issue is significant for logicians since (1) in a type theory, some assertions cannot be stated, e.g., "identity is identical to identity," within the theory consistently; and (2) theories that are capable of describing their own semantics can be developed within a type-free approach (Bealer, 1994). The latter highlights the fact that the system can be self-reflexive when built on an untyped basis. Moreover, Orilia (2000) mentions a *natural theory of properties* that Gupta and Belnap suggest while dealing with predication paradoxes. According to Gupta and Belnap's natural theory of properties, predication, "as a 'circular concept,' can be captured by circular definitions" (p. 267). That is to say, properties are *per se* circular, which is strictly forbidden in a type theory. Besides, Orilia (1991) insists that the classifications and principles in all these works have their own intuitive value, and their *raison d'etre* is independent of Russell's paradox.

The second one is adapted in intensional logic, an extended first-order logic that allows quantifiers to range over terms that may have extensions as their value. In other words, an intensional logic aims at formally representing intensional features that related to the distinction between *sense* and *reference* (Fitting, 2020). Bealer (1994) underlines that a type-free intensional logic has more representational power than a type-theoretical intensional logic, such as Montague grammar, because a type-free intensional logic can represent not only "our theoretical thought and talk - including our theoretical thought and talk about our theoretical thought and talk" (p. 165). Consequently, either to circumvent Russell's paradox regarding properties and predication, to construct a system whose complex properties and relations are built up from simpler properties, or to construct a formal system that can represent the sense-reference distinction, a type-free property theory regards properties as untyped in order to be meaningfully predicated of themselves (Orilia & Paolini Paoletti, 2020). Hence, freedom of types emerges in a single universe of discourse closed under a variety of operators.

Previously, we have shown that a typed theoretical framework has superiorities over a set theoretical framework; and a type-free theoretical framework over a typed theoretical framework. That is, a framework based on type-free theory

provides more abstraction so that it represents subtle expressions. The account starts with the fact that $\lambda\ x.xx$ is computable in type-free property theories. That means, properties can occur in both subject and predicate position (Orilia, 2000). Moreover, in a type-free property theory, entities can be modeled in various ways. For instance, the number zero can be represented as $\lambda\ f(\neg\exists\ x)\ (f(x))$, or as $\lambda\ fg\ \exists\ x((f(x) \wedge g(x))$,or as $\lambda\ fg\ \forall\ x((f(x) \rightarrow g(x))$, or as $\lambda\ f(\neg\exists\ x)\ (I^2(f,x))$, where where I is instantiation function (Orilia, 1991) (Cf. Orilia & Paolini Paoletti, 2020). All these representations can be typed.[19]

Replacing typed properties with untyped ones proposed more abstraction so that type-free theories play crucial roles in mathematics and natural language semantics as their foundations, in formal ontology as a handy tool for applications (Orilia & Paolini Paoletti, 2020; Orilia, 1991, 2000). Besides, untyped systems are crucial for computation. Cardelli and Wegner (1985, p. 473) state that

> As soon as we start working in an untyped universe, we begin to organize it in different ways for different purposes. Types arise informally in any domain to categorize objects according to their usage and behavior. The classification of objects in terms of the purposes for which they are used eventually results in a more or less well-defined type system. Types arise naturally, even starting from untyped universes. [$\cdots$] Untyped universes of computational objects decompose naturally into subsets with uniform behavior. Sets of objects with uniform behavior may be named and are referred to as types. For example, all integers exhibit uniform behavior using the same applicable operations. Functions from integers to integers behave uniformly in that they apply to objects of a given type and produce values of a given type.

Accordingly and counterintuitively, it is natural there be computational objects, which are unknown to us whether they actually denote entities, that have their own types (Galmiche, 1990). Thus, a type-free system based on symbolic expressions is obtained by a type theory interpretation (Galmiche, 1990). That is, an untyped system provides a level of abstraction where there are unlimited symbolic expressions; then, symbols turn into types, and types are executed in a type theory (Cf. Cardelli & Wegner, 1985, p. 473). Creating a new level of abstraction pertains to creating new types with untyped objects. That is tenable

[19] This is where polymorphism arises. So, polymorphism is a natural putcome of emplyig untyped units.

since even if we have no idea if the terms denote something, by interpretation, each typed term is associated with a specific expression of the logical theory; thus, a type-free theory can furnish a basis for typed constructions (Galmiche, 1990).

As mentioned, typed systems guarantee a consistent system, provide expected properties of data types and operations on objects, impose constraints to enforce correctness, and preserve precise and indented uses of the representations. However, untyped systems account for self-reference and contingency; in the end, they offer what type systems offer and more. The objects without content, then, protect losses in representations. That is to say, for instance, a yellow-trope should be represented in a contentless fashion, as it behaves differently in two distinct contexts. Speaking of, there may be several yellow-tropes –like number zero-that is why indefinite articles are used for tropes. Namely, there is nothing *the* x-trope. In the same vein, for instance, there can be several 'locomotions.' Each 'locomotion' has its own trope composition. Thus, different contentless units end up referencing the same trope name, and different contentful units can end up referencing the same trope composition name. Thus, similar to the famous example that 'the Morning Star' and 'the Evening Star' both designate the planet Venus but do not share the same meaning: the trope can have different meanings; viz., different types. As the types determine the allowable interactions, the machine could differentiate the intended usage of tropes by looking at their inner structure and external interactions. Therefore, tropes must be composed of simpler logically empty units (Cf. Orilia, 2000); in other words, tropes as types approach must be based on an untyped system.

3.5.2 An Untyped Theory

Here is a list of things essential for trope theory modification.

- Contentless abstraction of entities is required so that their *unrestricted* combinations are used to generate contentful entities.

- Self-reference must be allowable in a machine-understandable system so that the machine can design and describe interactions between entities in any context.

- The building blocks must be organized in different ways for different purposes to avoid losing any facts about the world.

- The machine should produce super- and subtypes within a machine-understandable system. That is to say, the machine can list super- or subtypes of, say, integer type.[20]

  - "Untyped universes of computational objects decompose naturally into subsets with uniform behaviour" (Cardelli & Wegner, 1985, p. 473).

The conclusion we reached hitherto is that the way for constructing machine discovery is in the power of an untyped system. As tropes are typed, so as extensional, we suggest a theory that is based on untyped computational and, at the same time, non-extensional ontological objects: *the Urtrope Theory*. The prefix "ur-" is borrowed from German that is an equivalent of the prefix "proto-" meaning that earliest form of, primitive, or original.[21] In the urtrope theory, tropes are compositions of urtropes, which are contentless. In this sense, urtropes are the building blocks of reality; thus, now we *do know* the building blocks of the relation-based ontology. Namely, primitive tropes are compositions of urtropes; complex tropes are compositions of urtropes and primitive tropes, or primitive tropes; all the entities are compositions of primitive and complex tropes.

To clarify the theory, consider an analogy to think of urtropes like the strings in particle physics. The analogy of string theory is not accidental, as the tangram, allotrope, and isomer analogies. We already know the atoms in an isomer; geometrical figures in tangrams. That is, the analogies used for introducing tropes are also extensional, just like tropes. On the other hand, as we will

[20] Those types are generated for semantic processes, not for human conception.

[21] The origin of the idea of employing this prefix to tropes is the set-theoretical notion of *urelement*.

see, the existence of a string is only known when it causes the emergence of a subatomic particle; that is, strings are non-extensional as urtropes. Strings are one-dimensional entities that construct subatomic particles; then, the string theory describes how the strings constitute the phenomena in the world. According to the theory, strings propagate through space. However, it is only when they undergo a particular mode of vibration and/or interact with each other (which vibrate as well) that they correspond to a particle with specific properties such as charge (Greene, n.d.).

Both the string and urtrope theory promise an intensional or non-extensional universe. In the string theory, vibrating strings twist and turn in various ways so that particles occur; in the urtrope theory, urtropes compose in various ways so that tropes occur. A string twisted in a particular way and vibrating with a frequency can correspond to a quark; another string twisted in the same particular way but vibrating with a different frequency can correspond to a photon; similarly, different compositions of urtropes give rise to different tropes. The analogy can be summed up with the following illustration, Table 3.1.

Table 3.1: An analogy between string theory and urtrope theory

Strings $\rightarrow$ quarks$\rightarrow$ atom$\rightarrow$ molecules$\rightarrow$ matter $\rightarrow \cdots \rightarrow$ world
Urtropes $\rightarrow$ tropes $\rightarrow$ entities $\rightarrow$ facts $\rightarrow$ context $\rightarrow \cdots \rightarrow$ world

From the computational perspective, urtropes are ontologically fundamental units that have no meaning at all. Consider them NULL; Unit or () or void in C++ in programming language; singleton set in set theory. No information can be taken from urtropes; there is no elimination; they just exist. For instance, consider the sequence of 0s and1: 01000011, which has no content at all. This symbol can represent the integer 67, the character 'C,' the 67th decibel level for a part of a sound, the 67th level of darkness for a dot in a picture, or an instruction to the computer, such as "move to memory," or else. Urtropes are like these 0s and 1s, and their combinations are like constructing a sequence.[22] In addition, we know that a letter can be represented with two different symbols

[22] Remember that we have not discussed a mathematical model suitable for urtrope and trope representations.

in two distinct representation schemas. For instance, 01011010 and 5A represent the letter 'Z,' whereas 5A can also be represented in 0s and 1s.

3.5.3 Interim Conclusion

Generating types of types was the reason for the road that took us to the urtrope theory. Thus, the primary motivation for introducing urtropes is to define types with non-extensional ontological things. In other words, the invention of urtropes is for generating and typifying extensional things.

That an urtrope is non-extensional means, it has no meaning at all. Their compositions, on the other hand, give rise to extensional things that have no meaning. Recall that an extensional thing gains its meaning only in a relational form. Furthermore, the urtrope theory does not specify propositional roles for urtrope compositions. That is, an urtrope composition does not assign static roles like a source entity or a relation in a context; a composition can be any of them. That is because the types can determine the interactions among the tropes that the machine can assign to tropes: there are axioms, in principle, that define the operations between urtropes and their compositions, viz., tropes. Hence, as Cardelli and Wegner (1985, 273) explicitly declare that "[t]ypes arise naturally, even starting from untyped universes[,]" constructing a machine ontology based on urtrope theory is absolutely legitimate.

3.5.4 A Machine Ontology as a Self-Representable System

The ontologies in Ontology 3.0 and the models similar to those are incapable of representing all possible interactions of all entities. As mentioned, an ER-model represents interrelated entities of a specific domain; thus, each ER model is a domain standardization. ER-models represent entities and relations in a very rigid, static manner. Similarly, each domain ontology is context-laden, which means all the entities have specific senses/behaviors/roles within the context. Entities and relations can be labeled differently in distinct domain ontologies, as they might have entirely distinct roles: the context changes, then the roles

change. Furthermore, we claim that even the upper-level ontologies (ULOs) are context-laden in the sense that entities are represented in the actualized domains.[23] That is to say, all the possible states of an entity cannot be represented by the totality of domain ontologies that are constructed on the same upper-level ontology. One may reject that upper-level ontologies are used as a schema so that the domain ontologies built on a ULO contain all the possible interactions of all entities.

We take the risk of repeating ourselves: each domain ontology is constructed for a purpose; when the purpose of the construction changes, the way the entities are categorized changes. Moreover, all these processes are for actualized domains. Even if domain ontologies are built on the same ULO, several other domains are there to be represented. We mean that the machine is destined for human annotations; we need to liberate humans and the machine from human annotations that specify semantic properties. Besides, a practical solution used in Ontology 3.0 and Web 3.0, merging ontologies is not enough to figure out the possible roles of an entity: there is a need to process the categories that categorize the entities. This is required since each categorization determines a class of roles and whose categorization also determines a new class of roles. Interactions between the classes expand the semantic properties of entities and provide a new class of possible interactions.

In Ontology 3.0, we humans need to build another ontology with other categories to process the categories. However, to process the categories of the latest ontology, we need to build another ontology whose categories categorize the categories of the latest ontology. Therefore, no ontology built by the Ontology 3.0 approach can cross the boundaries of the contexts, nor can it figure out higher and lower categories. If identifying upper and lower ontological categories is essential for getting rid of boundaries of contexts, then an ontology must be able to represent itself in its own ontological structure. Because, again, only then can it go out of context. This is nothing but the ontology itself becomes processable.

[23] For details of upper-level ontologies see section Ontology.

For instance, in order for the BFO to process itself, it needs to be represented in its own form of representation. However, the BFO cannot be represented in its representational form.

That an ontology can represent itself within its own representational form signifies that it is processable, just as the contexts it represents are processable. On the other hand, enabling an ontology processable is not sufficient for naming it a machine ontology. A machine ontology is also supposed to represent background knowledge; such a feature, however, requires infinitely many representations of the ontology. This is possible only if the representational form of the ontology is based on an untyped system; otherwise, the ontology is static, just like the ontologies in Ontology 3.0.[24] Hence, the urtrope theory is the foundation for a machine ontology.

Consequently, the urtrope theory enables an ontology to become a self-representable system. A self-representable system is a system that can represent itself in the same way in which it represents. As urtropes bring together infinitely many forms, the system can represent itself innumerable by staying within its representational form. The self-reflexive feature of the ontology, then, means that it is processable, and its processability enables possible semantic properties to be extracted with background knowledge. Lastly, note that the urtrope theory obliges a machine ontology to let it determine its own semantics (Cf. Bealer, 1994). In other words, syntax and semantics are on the same level of examination. So, the gap between operating in syntax and making sense of semantics will disappear. This is a feature of a creative and generative system.

[24] Thus, we are speaking of a dynamic picture of the world that cannot be attained by any typed system used in Ontology 3.0. Satıoğlu [Yargan] (2017) attacks the alleged feature of dynamicity in applied ontologies, where she defines *dynamicity* in a domain as the capacity to generate changes. In other words, a dynamic ontology means that the hierarchical structures are reorganizable according to a context. She argues that such ontologies are tailored for customers' needs and/or preferences, for the ontologies are designed for human-and-machine readability. Yargan criticizes Sowa (2010) for building a 'dynamic' ontology on the grounds of polycategorical systems and set theory. Both the syntax and the semantics are open to conceptualizations of ontology designers, which are devised in accordance with the application fields and the customer needs. Ontology 3.0 is not dynamic *per se* and cannot be dynamic in theory. Said that a machine ontology is not to be constructed single-use solutions or humans either: just let the machine itself know what the types are.

3.6 Ontology 3.0 vs. Ontology 4.0

The primary reason for constructing ontologies in Ontology 3.0 is categorizing entities of interest. Domain ontologies are built to represent entities and their interrelations of a specific domain, whereas upper-level ontologies represent overarching categories of entities and their interrelations. These ontologies are crucial for representing meaning in the machine with tools RDF, RDFS, OWL, and others.

Domain ontologies can only reflect the entities of the domain; that is, all the entities in a domain ontology have fixed roles/behaviors. For instance, a 'book' has different relations with other entities in a library ontology than a school ontology. In order to represent meaning in reality, it would be cumbersome to construct domain ontologies, as each domain ontology has its own reason for existence. So, entities and relations gain their meanings through annotations, which are the roles/behaviors specific to the domain in which they are defined. In other words, all the RDF or OWL classifications reflect just some aspects of entities and relations.

Can a collection of all these domain ontologies reflect the world so that the machine can understand, namely, make a reasonable connection between entities in Big Data? There is an affirmative answer that all these ontologies should be of the same form of upper-level ontologies, which we will visit in the next paragraph. The answer is negative if the domain ontologies are of different or not upper-level ontologies. As we have claimed so many times, Big Data offers an open-world view in which an entity can be one of its possible states. However, there may be some unrepresented roles/behaviors of entities and relations. Their new interactions emerge when a new entity is introduced in a context. Suppose such an interaction has not been represented in any domain, which is highly probable. In that case, the machine cannot relate the roles and make inferences in the collection of domain ontologies. In addition, none of the Semantic Web technologies can process metadata. In other words, if it is not represented separately, the machine cannot relate metadata and make inferences through them. So, there must be representations of interactions among annotations,

and another one for the interactions between those representations, and there can be a need for another. That is to say, not only should the interactions in a context be represented, but the interactions between the annotations of the entities should be represented. When this need is expanded to a collection of domain ontologies, there must be several other representations for metadata and representations for metadata of metadata. So, for instance, how can the machine figure out that a person with long hair can put their hair up with a pencil? The machine can discover such a role of a pencil when all these representations are implemented in the machine, which is definitely a Herculean task.

Besides, constructing upper-level ontologies cannot be a solution either. A design of a ULO differs in philosophical and implementation aspects. An ontologist must set their ontological positions beforehand; thus, they need to answer many questions, such as what an entity is, whether the ontology gives room for the events, whether there are abstract entities, and whether a time theory should be based on time intervals or time points. Meanwhile, the ontologist must ponder on theories of space and time, the relations between entities (which can be objects and processes, and both), dependency relations, independent entities, and alike. Determining particular philontological perspectives is crucial for identifying the uppermost categories of both reference and upper-level ontologies.[25] Nonetheless, employing these perspectives gives a solid representation of the reality depicted by humans, who have conflicting world views. If there are no abstract entities, what is π? Or, is π an independent entity? Which representation of π is correct? When the machine uses these depictions to 'understand' our reality, we need to wonder whether our depictions are suitable and sufficient for the machine to make inferences about our realities. As a solution, one can offer to build an upper-level ontology of upper-level ontologies. This, however, is implausible.

Another remark is on the formal languages we use for knowledge representation and implementation. Ontology 3.0 is capable of representing any entity, from

[25] See Satıoğlu [Yargan] (2015) for detail where she investigates the philosophical aspects of upper-level ontologies.

lollipops to Pegasus or metamorphosis. All these entities are represented with a set-theoretical approach and interpreted with model-theoretic semantics. Now, suppose that reality was classified and tagged, and we have had perfect domain ontologies for specific applications and *the* overarching domain-independent ontology. Would the mathematics we use for knowledge representation be good enough for automating reasoning?

Set theory, a —or *the*–logical foundation for mathematics, has been used as a medium for dealing with philosophical problems, for instance, by Russell, Quine, and Whitehead (Simons, 2005). Putting aside all the complexity issues, we need to highlight a severe pitfall in determining ontological structures with a set-theoretical approach. Set theory cannot reflect structural differences. Let us compare three different sets: the set of apples, the set of sheep on Ali Baba's farm, and the set of auto parts. Each collection has its own membership, causal powers, locations, and non-extensional identity conditions (Simons, 2005). That is to say, each has a different ontological treatment that is independent of mathematics. For instance, the ontological structure of auto parts has specific interrelations with each other, whereas a bunch of apples does not have such a complex interrelational web. Set theory is not capable of showing such a difference. Thence, we need to represent the structure of the collections, not their abstraction.

One may doubt whether classifying, determining the types of relations, and tagging the entities cannot mirror reality in the machine. An affirmative answer would be when we would have collections of entities of given types and in given contexts and/or locations. On the other hand, it is evident that we can list all such contexts neither in science nor on the Web due to their dynamic nature. Everything can be an element of a set, but not everything can be a part of a structure. When we utter about a context, we determine the entities by their roles/functions or properties, not by their names. Context building requires the selection of entities by their relations with other entities that play a role in the context. Set theory cannot provide a foundation for a self-organizational structure, which is pivotal in knowledge generation. Hence, set-theoretic construction cannot be a proper method of ontology.

Boden (2016, p. 39) mentions the tremendous labor of ontology design which is a present-day Herculean task as follows:

> One aspect of [...] lack of understanding is programs' inability to communicate with (learn from) each other because they use different forms of knowledge representation and/or different fundamental ontologies. If semantic web researchers can develop a highly general ontology, this Tower of Babel situation might be overcome.

However, Ontology 4.0 challenges this Tower of Babel approach. Unlike Ontology 3.0, Ontology 4.0 represents everything –objects, relations, and events– in terms of their semantic properties. In order to talk about autonomous machines, this approach is necessary. The dichotomy of entity and relations may be handy for humans to make sense of the world; however, this dichotomy does not help the machine understand the phenomena. So, from the machine intelligence perspective, understanding requires figuring out the semantic properties of both entities and relations. Consequently, the Semantic Web technologies cannot be used for such a representation.

Ontology 4.0 provides a solution to the open-world problem. Data 4.0 is data that has a new component, viz., semantic properties, along with its type and its value. The machine can know which semantic property of an entity is at work thanks to figuring out interactions among entities in the context. The typing rules determine the allowable interactions, so the machine can automatically assign entities' proper roles/behaviors in any context.

Computationally speaking, the mechanism of types of types in Ontology 4.0 allows the machine to figure out implicit interactions. In Ontology 3.0, types of types, namely metadata about metadata, is an extra work done after the context structure is determined. Illustrating types of types in Ontology 4.0, on the other hand, is inherited from the system. Everything can be decomposed to their urtrope and/or trope compositions so that all the possible interactions are unfolded. So, once the context is given, all the interactions, including super- and subtypes of entities, automatically emerge, and the typing rules confirm the appropriate ones. Lastly, the formalization tool of Ontology 4.0, viz., category

theory, is the theory of structure. It can solve all the pitfalls of the set-theoretical approach.

3.7 Conclusion

In this chapter, we purported that a trope theory, a philontology, was the most suitable ontological approach for machine ontology. We have modified trope theory to select properties according to a context and operate these properties. Then, we discussed that typifying the semantic properties was essential for processing, and we found that intuitionistic type theories were helpful to a certain extent. However, the fact that the process of typification must be conveyed by the machine automatically, trope and type theoretical approaches must be reconsidered once again. If the ontological basis is constituted upon non-extensional things, and the computational basis is constituted upon an untyped system, then a machine ontology can be possible. Consequently, we introduced *urtrope theory* as the ontological and computational basis for a machine ontology. So, we come to the point where we should find the appropriate formal theory that represents urtrope theory and executes the compositions. What formal system best suits the urtrope theory?

CHAPTER 4

AN ORCHESTRATION OF THE URTROPE AND CATEGORY THEORIES

This work claims that there must be a shift towards constructing machine ontologies, which serve for the *machine intelligence*. Machines and humans are of different categories, so as their intelligence and their intellectual agency (Zambak, 2014): the way humans interpret the world is different than the machines do or can do. However, studies in machine intelligence, or artificial general intelligence, are focused on how the machine can perform, just like how humans capture the phenomena, understand the world, communicate with their environment, employ reasoning and solve problems, and perform other intellectual operations. Nevertheless, the machine crunches symbols, the symbols that refer to things in the world. Thus, *being* and *data* must be orchestrated for the machine. For this reason, we started out to represent *phenomena into data*, that is nothing but *data within ontology*. More precisely, we aim to transfer entities in a formal realm where syntax and semantics share the same ground. As a result of our research, we found that if we want to establish a machine ontology, we need to define the world through relations and represent the building blocks as contentless beings. Consequently, we assert that formalization of the urtrope theory is the machine ontology that can realize the machine intelligence and solve the presented problems/issues of ISW 4.0s.

This chapter will show how the urtrope and category theories are brought together to create Ontology 4.0; in other words, we will show that *category theory* is the formal language of Ontology 4.0.

4.1 The Urtrope Theory and Representation

This part will explore how the urtrope theory represents the world. Let us recall its definition: Urtropes are the independent building blocks from which everything else is constructed. As urtropes are ["]ontologically["][1] independent, they *just* exist. So, there are entities (please take an entity as anything that is an urtrope composition) ontologically dependent on urtropes. Besides, from a computational point of view, urtropes present an untyped universe, yet their compositions present a typed universe. So, take that urtropes' being ontological objects and their being computational objects is one and the same thing.

Tropes, from philontological speaking, are the building blocks of reality; from the urtrope theoretical perspective, they are urtrope compositions. Accordingly, in a machine ontology that we propose, trope compositions define all the other entities. Recall that tropes are n-ary relations, $n \geq 1$ and that anything in the world is to be represented in terms of relations. Thus, anything represented in the machine is ultimately urtrope compositions. As the ontological aspect of the world in the machine is settled, then it comes to settle the primary reason for building a machine ontology: representing reality in the machine. To paraphrase, the formal representations of facts are trope representations. As the facts are nothing but relations; thus, a fact is a composition of compositions of tropes. One step further, a totality of facts gives a context; similarly, a totality of contexts gives an aspect of the world; lastly, the totality of the aspects of the world gives all the possibilities. Consequently, everything boils down to urtrope compositions. So, due to the realization of ISW 4.0s depends on the machine's ability to operate semantic properties, viz., tropes, we need to formalize this ontological and computational setting: We need a formal system that operates on urtropes so that tropes and their n-times compositions become processable.

[1] The intext air quotes are used for emphasizing that a machine ontology is spoken of, instead of a philontology.

4.1.1 A Formalization Tool for the Urtrope Theory: Category Theory

It should be started by saying that none of the formal systems that can be studied are specific to the urtrope theory. Formal systems have their goal to be solutions to problems in formalization, and accordingly, they all have their own ontological commitments. So, since there is no formalization system specific to the urtrope theory, we must choose the one that best fits the ontological commitments of the urtrope theory among many formal systems. That is, whatever we choose will not be specific to the urtrope theory, and we will choose according to whether the formal structure can represent the urtrope theory, not what it solves or what it aims in mathematics.

In investigating a machine ontology, recall that we concluded that everything that can be said about entities could be said by using relations. There is a formal theory telling a similar thing: everything can be said about objects can be said by using arrows. Such a reminiscent seems to signal that we have found the formal system we are looking for. Without further ado, this theory is called *category theory*, and we suggest that category theoretical formalization of the urtrope theory ensures a machine ontology that realizes ISW 4.0s.

We refer to category theory without mentioning other formal systems since it provides such a higher level of abstraction that it is regarded as the foundation of mathematics. Category theory (from now on CT) is the best abstraction tool to represent the urtrope theory since objects in CT are contentless and non-extensional. Also, CT has a relation-based approach: everything is represented from a relation-based perspective. This is precisely in line with the urtrope theory. Nevertheless, we will not use CT as it is. As we use CT to represent the urtrope theory, we will limit its employment with some axioms and rules. For example, type forming or concurrency rules require additional principles to CT. So, we will construct the formalization of urtrope theory with such additions to CT.

In the following, we will explore the birth of the theory and then give its mathematical definition along with some of its essential constructions. Later, we will talk about how the category theory can be used as a formalization tool for the urtrope theory. Please, bear with us till the end of this part since an explanation of an application of such an abstract theory suffers from a late closure.

4.1.1.1 An Apology for the Urtrope Theory formalized in Category Theory

As mentioned above, the implementation of the urtrope theory cannot admit the category theory itself. The introduction of CT and its usage in mathematics is proof oriented; for this reason, the primary purpose of the usage of CT is to bring different mathematical theories together and prove things in the best-known theory/theories. For instance, a complex topology problem can be proved in algebra, where handling the algebraic objects is more straightforward than topological ones. So, CT provides a sort of platform where different languages are translated into a common language in which communication can take place. However, as CT studies have been done for proof-oriented purposes in mathematics, using CT other than studying different theories together is not very common. That is why CT has no dominance in philosophy, computer science, modeling, information systems, and alike.

On the other hand, we will employ CT for its prowess in representation power, resulting from its higher level of abstractions. As mentioned above, CT and the urtrope theory have the same approach of representing things; however, they are not the same. CT has different ontological commitments/assumptions that have nothing to do with the urtrope theory and vice versa. Thus, there will be particularized axioms specific to the urtrope theory and accordingly to Ontology 4.0.

4.1.2 An Interlude: Abandoning Set Theories

Before we explore category theory, a few notes of caution are in order. We want to remind and highlight some reasons for abandoning formalizations of set theories and, accordingly, any formal system that depends on set theories. First of all, a set theory is not an appropriate tool for theory construction, which happens to permit a framework for any context. Vickers (2010) highlights that a set theory cannot offer a theoretical framework; rather, it offers contextual modeling, where a uniform ontological account of negation and universal quantification cannot be discussed. Moreover, Simons (2005) states that set-theoretical construction cannot be a proper method of ontology because a set theory has no natural interpretation as numbers do and because a set theory allows inappropriate attributions to properties, such as causal powers. Secondly, a set theory cannot reflect any structure of compositions. Simons (2005) criticizes set theory for its maximal promiscuous use of the term 'element,' which happens to combine arbitrary elements. An alleged economy of reduction of entities into sets causes detriments to ontologies: "everything can be an element of a set, but everything cannot be a part of a structure." Thus, any fact about a structure of a set cannot be exhibited from such an arbitrary collection; so, analogies cannot be driven, such as an analogy between the solar system and an atom is driven by their structure.[2] Thirdly, a set theoretical approach cannot formalize an untyped framework since everything is a set that is extensional in a set theory; thus, an untyped entity, an urtrope under our usage, cannot be a set.

Further, the very fundamental notion of a set is derived from the notion of elements, which are definite and well-distinguished objects of a collection. The fundamental relation in a set theory is the element relation, viz., being an element of, that necessarily requires a distinction between objects and this relation. On the other hand, we claimed that, for a machine ontology, there should be no ontological distinction between objects and relations. In addition to this, the element relation is inherently not compositional in nature (Healy & Caudell, 2006),

[2] That said, category theory is *the* science of structures.

on the other hand, the representation of the world in machine relies on compositions of relations. Finally, although some other reasons can be listed as well, type theories offer more abstraction, namely more representational power than set theories, and the paradoxes of set theories are eliminated in type theories.

To summarize, set theory is not an appropriate tool for implementing machine ontology. As a final note, since Cantor, set theoretical language has permeated into mathematics, as well as in philosophy, as a *lingua franca*, besides the fact that whether a set theory is the fundamentals of mathematics is controversial.[3] The purpose of this work, however, is not to discuss the foundation of mathematics or whether topos theory presupposes a set theory; instead is to provide a formalization that can implement urtrope theory into the machine.

4.2 The Birth of Category Theory

The basic concepts of category theory were born in the 1940s when Samuel Eilenberg and Saunders Mac Lane were working on the phenomenon of natural equivalences (Mac Lane, 1998; Kuś, Skowron, & Wójtowicz, 2019). These two mathematicians constructed some mathematical objects called *natural transformations*, *functors*, and *categories* (in chronological order) in order to study and describe different kinds of mathematical structures in terms of their allowable transformations (Awodey, 2006). Then, they applied their theory of natural equivalences to study more complicated problems of topological spaces using tools from abstract algebra. Later on, these auxiliary notions became important concepts. Then category theory (from now on CT) exceeded being a theory of natural equivalences and emerged as a conceptual framework that unifies various branches (Mac Lane, 1998), if not all of mathematics (Krömer, 2007).

[3] For instance,Feferman (1977), Bell (1981), and Hellman (2003) fiercely defend the superiority of set theory over category theory, whereas Lawvere (1964), Mac Lane (1986), and Lambek (2004) claim the opposite.
Blass (1984)'s work "The Interaction Between Category Theory and Set Theory" is a good starting point for comparing these theories.

4.3 Fundamentals of Category Theory

This part investigates the definition of a category and mention some crucial categorial[4] constructions. Then, to provide a flavor of CT, it examines the category of sets, `Set`. Let us fulfill these in turn.

A *category*, $\mathcal{C}$, consists of *objects* and *morphisms* and a *composition law* satisfying *identity* and *associativity axioms*. More precisely,

Definition 6 *A* category, $\mathcal{C}$, *consists of*

1. *A collection of objects of* $\mathcal{C}$, *denoted as* $Ob(\mathcal{C})$,

2. *A collection of morphisms, denoted as* $Mor_{\mathcal{C}}(X,Y)$, *or* $Hom(X,Y)$ *signifies that* X *and* Y *of the same category, where a morphism*[5] *corresponds to every pair of objects* X, Y *of* $\mathcal{C}$,
   A morphism f *of* $Mor_{\mathcal{C}}(X,Y)$ *is denoted as* $f : X \longrightarrow Y$.[6]

3. *For every object* X, *there exists a morphism* $Mor_{\mathcal{C}}(X,X)$, *which called the identity on* X, *and denoted as* id_X,

4. *Given any three objects* X, Y, Z *of* $\mathcal{C}$ *and morphisms* $f : X \longrightarrow Y$ *and* $g : Y \longrightarrow Z$, *there is another morphism which is a composition of* f *and*

[4] Echoing Goldblatt, we use "categorial" instead of "categorical" as the former emphasizes the intended use.

[5] Morphism has various names: Map, mapping, functions, transformation, and arrow. In order not to evoke any set theoretical atmosphere (since the term 'function' is a reserved term in the set theory), we disagree with using 'function' in CT. The term 'map,' according to us, is a generic term, and the term 'transformation' has problems, such as the idempotent morphism does not transform the state of a dynamic system. Under our usage, the term 'morphism' is a categorial generic term that refers to arrows, functors, and natural transformations. An arrow is a morphism between two objects of a category; a functor is a morphism between two categories, and a natural transformation is a morphism between two functors. As such, morphism is used interchangeably with arrow, functor, and natural transformation.
NB: A relation is a mapping with sense.

[6] A caveat to this notation. $f : X \longrightarrow Y$ does not mean that the elements of X are mapped to the elements of Y according to some rule called f. Rather, it means that there are two objects X and Y, and a 'morphism' called f between them: the morph of X is represented in Y. This is due to the fact that the primary intentions of introducing category theory were to offer a language for stating certain mathematical concepts in *abstract nonsense* (Biss, 2003, p. 577). Thus, all the known mathematical notions similar to the notions used in category theory must be set aside, especially the set-theoretical notions, such as 'function,' 'being an element,' and alike.

g, such that $f;g : X \longrightarrow Z$. This composition, call h, is also a morphism of $\mathcal{C}$.[7]

In order this quadruple to be a category the following laws must be satisfied:

1. *Associativity: For all objects X, Y, Z, and W, and all morphisms $f : X \longrightarrow Y$, $g : Y \longrightarrow Z$, and $h : Z \longrightarrow W$, the equality $f;(g;h) = (f;g);h$ holds.*

2. *Identity: For all objects X and Y, and all morphisms $f : X \longrightarrow Y$, the equalities $id_X;f = f$ and $f;id_Y = f$ hold.*

This is the definition of a category, which can be used for constructing or examining a category.[8] In the following, categories of some mathematical structures are given for gaining familiarity, which are taken from Awodey (2006), Mazur (2008), Adámek, Herrlich, and Strecker (2004), Biss (2003), and Roman (2017).

The category `Mon` of monoids, whose objects are monoids and whose arrows are monoid homomorphisms,

The category `Grp` of groups, whose objects are groups and whose arrows are group homomorphisms,

[7] The morphism composition can be showed as $g \circ f$, (f, g), or $g \cdot f$. We use diagrammatic order $-f;g-$ to ease reading and make distinct notation than the classical orders.

[8] In order to specify a category, the following must be specified (Lawvere & Schanuel, 2009, p. 166):

1. The objects,
2. The morphisms,
3. Which object is the domain of the each morphism,
4. Which object is the codomain of the each morphism,
5. Which morphism is the identity of each objects,
6. Which morphism is the composite of any two composable morphisms.

After these are specified, the accurate compositions of morphisms must be verified. Further, the following laws must be checked.

1. The identity law
2. The associativity law

If all of these are satisfied, then the construction is a category.

The category `AbGrp` of abelian groups, whose objects are abelian groups and whose arrows are group homomorphisms,

The category `Top` of topological spaces, whose objects are topological spaces and whose arrows are continuous functions,

The category `SmoothMan` of manifolds with smooth maps, whose objects are manifolds and whose arrows are smooth maps,

The category `Rng` of rings, whose objects are rings with unit and whose arrows are ring homomorphisms,

The category `Field` of fields, whose objects are fields and whose arrows are ring embeddings,

The category `Set` of sets, whose objects are sets and whose arrows are functions,

The category `Poset` of groups, whose objects are partially ordered sets and whose arrows are monotone functions,

The category `Rel` of relations, whose objects are sets and whose arrows are binary relations; viz., cartesian products,

A deductive system $\mathcal{T}$, whose objects are formulae and whose arrows are proofs,

The category `Aut`, whose objects are automata and whose arrows are simulations.

4.3.1 The Category of Sets

The categorial framework can be applied to any branch, and the subbranch of mathematics is just exemplified above. The most crucial point to consider while applying this framework is figuring out how a theory's concepts and their combinations can be expressed in terms of morphisms and compositions of morphisms (Lawvere & Schanuel, 2009). In this part, we will explore the category of sets, denoted as `Set` or $\mathcal{S}$. The purpose of choosing this category is twofold.

The first is its overt recognition: set theory as a mathematical branch is familiar even to a high school student. As such, it is an example of a quick opt, and there would not be any extra effort to contemplate the category theoretical properties and constructions. The second one is that Eilenberg and Mac Lane, the forbears of category theory, introduced categories as an auxiliary purpose to construct concepts of functors and natural transformations that are in set theoretical background and language (Marquis, 2021).

The *category of sets* consists of *sets* and *functions* and a *composition law*, and satisfies *identity* and *associativity axioms*. More precisely,

Definition 7 *The* category of sets, `Set` *or* $\mathcal{S}$*, is a category of which*

1. *Objects are sets,*
2. *Arrows are functions between sets,*[9]
3. *Composition of arrows are composition of functions,*
4. *For every set* X*, there exists a function, called the identity function on* X*, and denoted as* id_X*,*

Associativity and identity laws are satisfied, since compositions of functions are associative, and for any function $f : X \longrightarrow Y$*, the equalities* $id_X; f = f$ *and* $f; id_Y = f$ *hold.*

Informally speaking, contrary to a set theory, the elements of sets are not of concern but rather the functions between the sets in $Ob(\texttt{Set})$. However, this does not mean that the elements of a set cannot be shown. As the key of CT, everything is expressible in terms of morphisms so that the elements of a set are to be shown by morphisms.[10] Adequately, there are other facts about sets,

[9] In some settings, an arrow of `Set` can be defined as a triple $(X; f; Y)$, where X and Y are sets and f is a function from X to Y. As such, other morphisms, such as identity arrow $(X; id_X; X)$, are indicated as triples. However, this notation becomes more complicated when indication further categorial constructions.

[10] The work is an intricate one, so we will not fully recount it here. Suffice to say that the very first

such as the empty set, set intersections, or set products. All of these, some of which we will see below, are translated into a categorial framework; for instance, the empty set is a particular object of `Set` called an initial object,[11] and any singleton is a terminal object of `Set`.

4.4 Further Definitions and Some Categorial Constructions

Hitherto, functors, natural transformations, or initial objects were mentioned without their definitions. In this part, we will examine the categorial concepts from the beginning, then show the important constructions less formally.

4.4.1 Morphisms

The networks of relations determine the entities in the world, says Rovelli (2021), as the network of morphisms determines the mathematical objects in category theory. A category is characterized *only* by its morphisms. We can even say that the networks of morphisms can replace the objects (Mazur, 2008). Just as relations in real life help us understand and learn more about the phenomena, CT helps us understand and learn more about mathematical objects by looking at how they are related to each other. Further, it is legitimate to claim that morphisms are the primitives of CT as they are the only information source about any mathematical object. In other words, the collection of morphisms to and from the objects and the algebra of composition of morphisms specify the objects and pave the way for discovering properties of objects (Krömer, 2007).

thing to do is to transform the set-theoretical concept of 'element,' which is considered a singleton set. That is, any element of X is to be represented as a singleton set. Then, when a function that maps each singleton set to its corresponding element in X is found, it can be represented in terms of CT. To make the long story short, the elements of a set can be expressed in terms of morphisms, which are *points*.

[11] Indeed, the empty set is *the* initial object of `Set`. This point has the utmost importance in CT, which will be explained in the following paragraph.

4.4.2 Objects

The objects of a category play a minor role, as a category is characterized by its morphisms, not by its objects. The properties of an object cannot be discovered by studying the object. Thus, the only thing of concern about objects is their existence, and the only thing of concern about categories is "the list of morphisms between all pairs of objects and of the rules for composing these morphisms" (Biss, 2003, p. 576). Well then, what is an object when "no object is known in complete isolation from others"? (Peruzzi, 2006).

Eilenberg and Mac Lane admit the secondary role of objects, which can even be omitted entirely (Marquis, 2021). As a matter of fact, there is no explicit definition of it, but rather an implicit one: objects are characterized by the morphisms going in them and/or the morphisms coming out of them, and up to (unique) isomorphisms (Marquis, 2021). Krömer (2007) claims that the term "object" has traditional uses outside mathematics, so its connotations in everyday language and philosophical discourse must be surveyed. The most familiar one is set theoretical use, which often goes with the predicate "is an element of." Or, in everyday language, an object is an entity with an extension. However, in CT, a mathematical construction is an object that "shrinks to a point which cannot be penetrated and of which one knows only the traces left by its interaction with other objects" (Krömer, 2007, p. 218). Only through the means of the constitution can an object be characterized; thus, we should unlearn the traditional semiotic tenets. As we could not refer to an object but rather know it from the traces of its interactions, an investigation of an object is open-ended; in other words, such traces give only some aspects of the object. As such, any information about an object is present in the totality of the morphisms arriving at, and departure from the object (Krömer, 2007). Now, we arrive at one of the most crucial aspects of CT: the morphisms arriving at and departure from an object enable process objects without knowing their internal structure. It should be clear that the objects of the theories, e.g., a set theory, are not the objects of CT; in other words, the theories are the objects of CT, and the objects of the theories are of no concern in categorial constructions (Krömer, 2007). Then,

CT deals with the objects of the theories without penetration, namely, without tearing them apart or interrupting them (Lawvere & Schanuel, 2009).

An object of a category is characterized in categorial terms. A vital challenge of this process is that objects are defined without referring to their internal structure. For instance, in the category of sets, the empty set must be defined without referring to its set-theoretical structure, or, say, the empty set must be defined without referring to elements. As highlighted before, an object is characterized by the morphisms arriving at and/or the morphisms coming out of it, morphisms of the respective category. There are *universal* notions in CT, which are determined by specific properties of morphisms that satisfy certain properties of the category at hand. For the case of empty set in `Set` an initial object, which is a universal construction, is of `Set` refers to the empty set. Hence, the objects of theories can be described in terms of universal constructors, such as limit, pushout, and others.

In addition to all, consider the illustration. In the category of sets, the sets, namely the objects of the category, may have precise ways of affecting each other, special characteristics, and are internally attached together, such as a set and its power set, the empty set, and the intersection of two sets, respectively. The arrows of `Set` not only allow construction and study of such sets and operations but also enable comparing these sets and operations without penetrating the sets. Moreover, by applying specific functors, we can even study and compare these sets without interrupting them; for instance a specified forgetful functor between `Top` and `Set` can be used for comparing two sets of `Set`.

The implicit definition of an object in CT reads, "objects are characterized [...] up to (unique) isomorphism." In other words, almost all mathematical objects can be described in various ways, some of which are equal up to isomorphism (Biss, 2003). This statement is directly related with the use of indeterminate article for categorial constructions; e.g. *a* limit or *a* terminal. Consequently, for instance, there is no object as *the* natural numbers, rather there is *the concept* of natural numbers. Mazur (2008) defines *natural numbers* as an object is an initial object of the Peano category; the concept of natural numbers, on the other

hand, unambiguously given in each category. In other words, each category as a context determines how this concept will be interpreted in a definite way. Thus, theoretical constructions are unique for the theory but not for CT: objects can have other representations with different categorial contractors. For instance, there is "the" unit element for multiplication in algebra, but there is "a" unit element for a categorial construction for multiplication in algebra in CT. Put another way; a given object can be substituted by its isomorphic object in any situation that it is categorially expressible (Krömer, 2007).

Said all, a caveat is in order. The identity criterion is fundamentally distinguished between category theory (CT) and set theory (ST). In ST, mathematical objects are identical if and only if they satisfy the axiom of extensionality. In contrast, in CT, as we mentioned just above, the mathematical objects are identical, or say equal, up to unique isomorphism. A crucial difference between CT and ST arises from thinking of mathematical objects as types instead of defining them uniquely and giving their reference directly (Marquis, 2021). Mathematical objects as types can be defined in different senses in different categorical contexts, and each sense is taken as a term. Moreover, the type-term relation cannot be taken as a membership relation. So, again, there is nothing like *the* natural numbers, but *the concept* of natural numbers:[12] $\mathbb{N}$ is a type of initial object type in the category of Peano or the pairs of (ω, F) in the category of topological spaces.[13]

Here are some counterintuitive facts about CT. For instance, some characteristics mentioned earlier of CT admit to dynamic ontologies, in which an object and an arrow can be equal up to an isomorphism. That is, a complex construction in one category has its own ontological status, but this is not written in stone: it can shrink to a simple object in another category (Krömer, 2007). Alternatively, being property is not an absolute ontological status; in one categorial setting,

[12] Similarly, there is nothing like *the* set theory. Instead, there is *a* set theory and *the* category of sets.

[13] The explanation of ω and F is omitted for the sake of simplicity. Those who want to learn more about limit spaces and filter spaces and how the natural numbers are defined in these structures can refer to Asperti and Longo (1991), pp. 55–58, pp. 103–104.

it might have characteristics of a property, and in another setting, of an object. Or, consider this: morphisms of a category can be considered objects of another category, or objects of a category can be considered morphisms of another. These seem untenable at first glance, but in CT, they all sum up to a "strategy of relativization" that paves the way for dynamic representations. Krömer (2007, pp. 219–220) expounds all these as

> [G]iven objects can be considered in extension in one category and as points in another; in some cases they can be categories themselves and simultaneously objects of a category and arrows of another and so on, where in every case another type of "rapports entre les choses" is accentuated.

Thus, ontological statuses of mathematical entities are not rigid but rather varying, and they hinge on the chosen level of thematization. The distinction between objects and morphisms is flexible since a relativization strategy determines the most appropriate construction. So, what are the natural numbers? Although there is *the concept* of natural numbers, there are various natural numbers constructed in different categories; they can be a morphism, a category, or an object.

4.4.3 Functors

Although the term 'functor' is used several times in this work, we sufficed to define it as a morphism between categories. Now, we will give its mathematical definition and discuss its importance in theory.

Definition 8 *A functor,* $F : \mathcal{C} \longrightarrow \mathcal{D}$ *, is a morphism between categories* $\mathcal{C}$ *and* $\mathcal{D}$*, such that, for any object* X *of* $\mathcal{C}$*, there is a well-defined* $F(X)$ *of* $\mathcal{D}$*, and for any arrow* $f : X \longrightarrow Y$ *of* $\mathcal{C}$*, there is a well-defined morphism* $F(f) : F(X) \longrightarrow F(Y)$ *between corresponding objects of* $\mathcal{D}$*.*
The compositions and identity morphisms of $\mathcal{C}$ *must hold the corresponding compositions and identity morphisms of* $\mathcal{D}$*.*

More precisely, let X, Y and Z be objects, and $f : X \longrightarrow Y$ and $g : Y \longrightarrow Z$ be arrows of $\mathcal{C}$. If F is a functor between categories $\mathcal{C}$ and $\mathcal{D}$, the following must be hold.

1. $F(f : X \longrightarrow Y) \;=\; F(f) : F(X) \longrightarrow F(Y)$,

2. $F(f;g) = F(f);F(g)$,

3. $F(id_X) = id_{F(X)}$

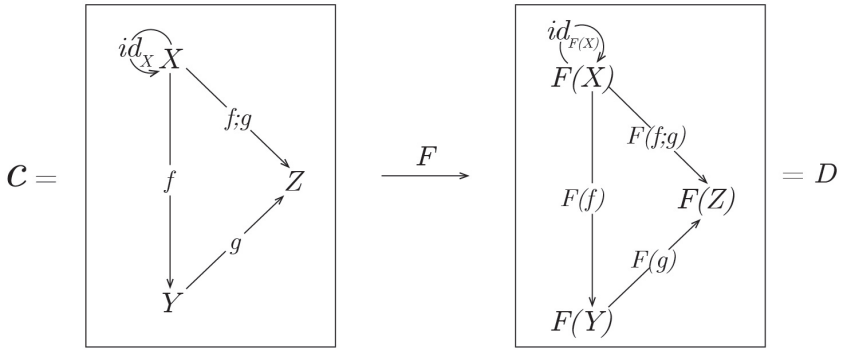


Figure 4.1: F preserves composition and identity.

CT is often labeled as the theory of functors –the structure preserving morphisms between categories– via which structures are studied (Krömer, 2007). Indeed, while Eilenberg and Mac Lane were working on natural equivalences, they came up with the idea of functors long before the concept of categories. However, in order to define them formally, the mathematicians needed to introduce categories before the functors. There are various types of functors, such as identity, forgetful, faithful, representable, amnestic, embedding, homology, cohomology, or homotopy K-theory (Marquis, 2021). Namely, the studies of CT can be boiled down to typifying functors. Nevertheless, what makes them so unique?

Functors are special since they reveal and/or transfer and/or interpret *knowledge* between categories. For instance, they can be used as a vehicle to transfer problems from one branch to another, where solutions are often easily reached. The facts of abelian groups help solve some problems in topological spaces, and the work is conveyed by cohomology -a kind of functor- reveals the structural relation between these categories (Adámek et al., 2004; Marquis, 2021). Functors also provide integrity of structures, as Spivak (2015) utters "a functor *between* two categories is also required to align the multifaceted relationships that exist *within* the categories" [italics in the original]. More generally, functors are

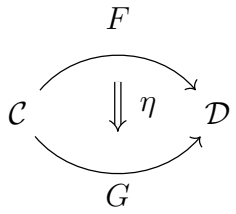
responsible for revealing how different kinds of structures are related to one another; more specifically, they are to offer new knowledge about the structures by forming and layering them. In other words, they represent one theory in another: any structural constraints expressed in a category can be translated into another category by functors. Once a category is known, it can be used to reveal the facts of an unknown category via a systematic study of the functors between them (Biss, 2003): functors represent one theory (category) in another (category), such as group theory can be represented in topology by creating functors. Since the structure of the known category is preserved through the functors, the structure of the unknown or little-known category becomes elucidated. Hence, the known, the unknown, and the little-known among various categories are used for understanding the categories themselves.

For the time being, make do with the following fact, which we will visit in the coming parts. In the programming realm, functors correspond to the type operators in generic programming, that is, indeed, functorial programming, in which algorithms are written in terms of types (Harper, 2016). Thus, types are specified functors and categories express type compositions.

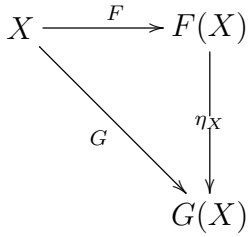
4.4.4 Natural Transformations

Natural transformations are structure-preserving morphisms between functors. According to this definition, it seems that functors are objects, and the natural transformation between them is a morphism. Using the very basics of categorial algebra – of objects, arrows, and functors-the definition of natural transformations was introduced, even if the first intension of the work of Eilenberg and Mac Lane was to define natural transformations formally. That is, a natural transformation is a special kind of morphism between morphisms. More precisely,

Definition 9 *Given two categories* $\mathcal{C}$ *and* $\mathcal{D}$*, with two functors between them,* $F, G : \mathcal{C} \longrightarrow \mathcal{D}$ *, there is a morphism, called a natural transformation,* $\eta : F \longrightarrow G$ *.*



For each object X of $\mathcal{C}$, there is a morphism $\eta_X : F(X) \longrightarrow G(X)$ of $\mathcal{D}$. Thus, the following commutes:



The power of natural transformations lies in their ability to preserve structures. The focus is on the *structures* of functors, a novel idea for non-categorially oriented minds. To unpack the idea that morphisms have structures, consider that a natural transformation reveals the structural relation between the functors by forming and layering them. Functors are objects whose characteristics are determined by natural transformations and are to be processed just like the objects of a category. Nevertheless, natural transformations do more than provide means for thinking and analyzing the functors: they provide myriad interpretations from multiple perspectives. That is, they can also offer many studies on the categories in question by providing different levels of abstraction. This feature is the supreme power of natural transformations.

Definition 10 *The category of functors,* $\texttt{Fun}(C, D)$ *or* $\mathcal{D}^{\mathcal{C}}$ *has as objects the functors,* $F : \mathcal{C} \longrightarrow \mathcal{D}$ *, and as morphisms the natural transformations between the functors,* $\eta : F \longrightarrow G$ *.*

4.4.5 Duals

Another concept that is of utmost importance in CT is duality. In its simplest yet most straightforward explanation, a *dual* is obtained by reversing the directions

of all morphisms of a construction. The result of such a simple modification is, indeed, enormous. Adámek et al. (2004, p. 12) praise the duality in these words: "[...] in category theory the "two for the price of one" principle holds: every concept is two concepts, and every result is two results." Healy (2010, p. 494) states, "half the theorems of category theory are obtained for free." Well, what has a dual? A category has a dual when the direction of arrows is changed, in other words, $\mathcal{C}$ and $\mathcal{C}^{op}$ have the same objects and arrows, yet they have reverse directions. A property has a dual; for instance, the dual of an initial property is a terminal property. A statement also has a dual, such that if it is true in $\mathcal{C}$, then the dual of the statement is true in $\mathtt{C}^{op}$. Finally, the Duality Principle for Categories states that whenever a property P hold for all categories, the property P^{op} holds for all categories.

The duality principle allows us to move between the duals without loss of information and simultaneously investigate the same structure from different constructions. For instance, the pushout diagram allows us to examine a structure as a bottom-up construction, and the pullback diagram does it as a top-down construction (Neuman & Nave, 2008).

4.4.6 Universal Properties and Constructors

Category theory helps us understand and learn more about mathematical objects by investigating how they are related to each other within a category, making analogies between different categories, investigating facts about a category, probing the more complicated constructions, and many others. Additionally, category theory can be thought of as a language describing similar phenomena, or properties, occurring in entirely different mathematical theories. The categorial definitions offer abstractions over mathematical theories to study the similarities and relations among distinct mathematical structures and theories. Such abstractions enable a concurrent investigation of similar properties in different mathematical settings. For instance, the property of product is found in many theories, e.g., sets, groups, and vector spaces. Although a product construction is different in the mentioned theories, the property of product stays

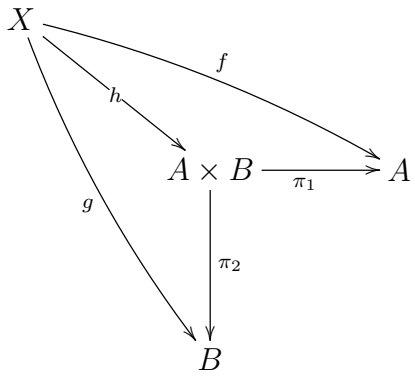
the same; it is CT that studies such properties, namely *universal properties*, and speaks of such *universal constructions*. Let us unpack universal properties in what follows.

The concept of universal property is no less important than the foundational concepts of category, functor, and natural transformation. However, universal properties are not particular to CT; for instance, product is also considered a universal property in algebra and set theory. But, the abstraction of universal properties is *particular* to CT: CT studies universal properties to point out similarities between different mathematical theories, some of which may seem totally irrelevant. In other words, universal properties, such as product, tensor product sum, or quotient, have their own constructions in different theories, and they all behave similarly in those theories; what CT does formalize such similar behaviors, or say, provides a template of a universal property, upon which properties of other objects can be recognized. Now, let us explain universal property with the following example.

As mentioned before, product is a universal property in mathematics. In set theory, it is defined as the set of ordered pairs, $(A \times B) = \{(a, b) | a \in A \text{ and } b \in B\}$; in group theory it is defined just like in set theory with further group theoretical characteristics, such as the operations in each group is preserved component-wise in the product; such that, G is a group with "$\circ$" operator, and H is a group with "$\star$" operator, $(G \times H)$ is the ordered pairs (g, h) and binary operation is defined as $(g_1, h_1) \cdot (g_2, h_2) = (g_1 \circ g_2, h_1 \star h_2)$. Product can be constructed in another mathematical theory as well. Said that, the concept of product is abstracted in CT, and it is defined as a template ready to be applied in any mathematical theory. So, the property of product is constructed in the most general form. As everything in CT is defined in terms of morphisms, product is not an exception. There must be a unique arrow that defines product, such that the construction of product with such a unique arrow must be true in any category.

Definition 11 *A product of two objects A and B of any category $\mathcal{C}$ is an object of $\mathcal{C}$ with two arrows, generally called projections, $\pi_1 : A \times B \longrightarrow A$ and*

$\pi_2 : A \times B \longrightarrow B$ *such that for any object* X *of* $\mathcal{C}$ *with* $f : X \longrightarrow A$ *and* $g : X \longrightarrow B$ *, there is a unique arrow* $h : X \longrightarrow A \times B$ *such that* $h; \pi_1 = f$ *and* $h; \pi_2 = g$*, and such that the following diagram naturally commutes:*



So, the product of two objects (e.g., sets, groups, and topological spaces) is specified by the property. What is 'natural' in this composition is that if h corresponds to f and g, and $h_1 : Y \longrightarrow X$ is an arrow, then $h_1; h$ corresponds to $h_1; f$ and $h_1; g$, so that h_1 and h are isomorphic. Moreover, suppose that there is an object P with arrows $\pi_1 : P \longrightarrow A$ and $\pi_2 : P \longrightarrow B$. For any pair of arrows $f : X \longrightarrow A$ and $g : X \longrightarrow B$, there is unique arrow $h : X \longrightarrow P$, which makes the diagram commute (nLab authors, 2021c). And consequently, $A \times B$ and P are equal up to the unique isomorphism. Thus, product is uniquely determined by categorial terms.

That said, the existence of isomorphism is necessary but insufficient for the constructed object to be a universal property. Nevertheless, before moving on, let us give the categorial definition of isomorphism here.

Definition 12 *Let* X *and* Y *are objects of* $\mathcal{C}$*. They are called isomorphic if there is a morphism* $f : X \longrightarrow Y$ *and* $g : Y \longrightarrow X$ *of* $\mathcal{C}$*, such that* $f; g = Id_X$ *and* $g; f = Id_Y$*,* f *and* g *are said to be isomorphism.*

Accordingly, the equality of two objects of a category is studied by determining if there is a specific isomorphism between the objects. So to say, from the definition above, X and Y are equivalent objects of $\mathcal{C}$; from the product example, $A \times B$, X, and P are isomorphic, or equal objects, when all above-stated arrows

commute. Consequently, universal properties are always unique up to isomorphism. A caution is here: there are some universal properties that do not exist in some theories; for instance, partial order categories without smallest and largest objects do not have universal properties. Then, let us turn back to the issue of why isomorphism is not sufficient for the existence of universal property. For the isomorphism to be 'natural,' universal properties are formulated by 'universal morphism.' When objects A and B are universal objects (this is another way of telling that these objects have universal properties), then there is a unique arrow between these two that corresponds to P. Indeed, this unique arrow, called *universal morphims*, emerges the property. Hence, a universal property is a property that is satisfied by a universal morphism. More precisely,

Definition 13 *A universal morphism from A to F is a pair (X, h), where X is an object of $\mathcal{C}$ and h is an arrow $h : A \longrightarrow F(X)$, such that for every arrow, $g : A \longrightarrow F(Y)$, of $\mathcal{D}$, there is a unique arrow of $\mathcal{C}$, $f : X \longrightarrow Y$ with $g = h; F(f)$.*

Thus, objects with a universal property are defined up to a unique isomorphism in the appropriate categories. This points out that some of the objects of a category are characterized by means of universal properties. Further, the definition of a universal object also defines a universal property, and a unique morphism indeed defines a property. Here comes new information about such objects: a universal object is either an initial or terminal object. Alternatively, if an object satisfies either initial or terminal property, it is called universal. Strictly speaking, a universal object is an initial or terminal object depending on the theorem.

Definition 14 *An initial object,A, of $\mathcal{C}$ is an object with the property that given any object X of $\mathcal{C}$, there is a unique arrow $i_X : A \longrightarrow X$ of $\mathcal{C}$.*

Definition 15 *A terminal object, A, of $\mathcal{C}$ is an object with the property that given any object X of $\mathcal{C}$, there is a unique arrow $i_X : X \longrightarrow A$ of $\mathcal{C}$.*

When the definitions are examined closely, one can recognize that initial and terminal objects are duals of each other. Thus, a universal object is a dual of another universal object. A category might have more than one initial/terminal object. Yet, there is a unique isomorphism between initial/terminal objects, and what matters is not the objects by themselves but the specialty of the uniqueness of the isomorphism. As we noted before, a category does not necessarily have a universal object or objects. For instance, the category of topology has the empty space as *the* initial object and the one-point space as *the* terminal object; the category of rings has *an* initial object $\mathbb{Z}$, yet has no terminal object; the category of fields has neither an initial nor a terminal object (Biss, 2003, p578).

A *universal construction* is the definition of a universal object. To be more precise, it is a construction in the sense that the object's existence must be shown first, and then the object is characterized by a universal property. So, while constructing the object, the definition of the universal object is shown simultaneously. The importance of universal constructors is their role in providing conceptual unification of different mathematical theories (Peruzzi, 2006). To name some universal constructors, CT includes representable functor, adjoint functor, limit/colimits, end, Kan extension, and dependent sum/dependent product (nLab authors, 2021c).

We have spoken of universal properties and constructions at length since they are as essential as the basic notions of category theory due to their tremendous benefits. Firstly, universal properties are everywhere in mathematics. As such, once its abstract properties are illustrated, a universal property provides knowledge about the objects that satisfy it. In other words, universal properties transfer the knowledge of an object –satisfies a universal property- in a theory into an object – also satisfies the same universal property- in another theory, so there would be no need to study objects further. For instance, the universal property of initial object is known to be satisfied by the empty set in set theory. What we know about the empty set can be transferred to an initial object of any category at hand; without proof, we say that "such and such characteristics belong to this object," as we know that characteristics from the empty set. For instance, an initial object of the category of set is the empty set, whose proper-

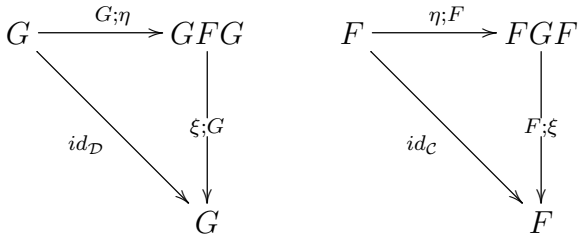
ties in the categorial sense are the same as the initial object of the category of matrices. This benefit brings us to the second one: as universal properties define objects up to a unique isomorphism, it suffices that two objects satisfy the same universal property to show that they are isomorphic. Thus, there would be no extra labor for investigating isomorphism within a theory and among theories. For instance, when that the empty set of `Set` and the ring of $\mathbb{Z}$ of `Ring` are both initial objects is known, then it is also known that they are isomorphic. Accordingly, and thirdly, objects can be studied from various perspectives thanks to universal properties. That is to say, a construction can be defined and investigated in many different ways. For instance, open or closed sets, neighborhoods, convergent filters, or closure operations can define topological spaces (Adámek et al., 2004). The next benefit is that universal properties save us from dealing with, or getting lost in, details of a construction. For instance, suppose that two objects and an operation between them are defined in a theory. If these satisfy the universal property of product, the new object, namely the construction, becomes free of complicated and messy proofs. That is, if a construction at hand satisfies a universal property, then the details of the construction can be forgotten, and/or using universal properties make proofs shorter. Finally, universal properties bolster up defining new constructions. For instance, the product of sets satisfies the universal property of products, and its definition in a categorial setting is just the implementation of this universal property (Cf. nLab authors, 2021c). Further, by choosing a category whose universal properties are already acknowledged, a totally new category can be constructed.[14] That is, a new category can be constructed with the aid of universal objects in categories already known at full length. Consequently, universal properties and constructions help to "detect analogies and connections to familiar fields, to organize [a] new field appropriately, and to separate the general concepts, problems and results from the special ones, which deserve special investigations" (Adámek et al., 2004, p. 12).

[14] This is what we can do while formalizing urtrope theory.

4.4.7 Adjoints

In the previous part, it is noted that adjoint functors are universal constructors that provide conceptual unification. Of various types of functors, adjoints have a crucial place for CT; even they can be called "the cornerstone" of CT (Marquis, 2021). The cruciality is twofold: adjoints help investigate not only mathematical theorems but also category theory in itself. Let us start with its formal definition before a journey to investigations with adjoints.

Definition 16 *Let* $F : \mathcal{C} \longrightarrow \mathcal{D}$ *and* $G : \mathcal{D} \longrightarrow \mathcal{C}$ *be functors.*[15] F *is the left adjoint to* G *and* G *is the right adjoint to* F*, if there exists natural transformations* $\eta : id_{\mathcal{C}} \longrightarrow FG$ *and* $\xi : GF \longrightarrow id_{\mathcal{D}}$ *,*[16] *such that the compositions are the identity natural transformations:*



Peruzzi (2006, p. 425) explains the first part of cruciality of adjoints as

> What is central to any foundational project proper is the role of universal constructions that serve to unify the different branches of mathematics, [...] Such universal constructions are best expressed by means of adjoint functors and representability up to isomorphism.

Moreover, Awodey (2006, p. 179) moves the state of cruciality to another level:

> The notion of adjoint functor applies everything that we have learned up to now to unify and subsume all the different universal mapping properties that

[15] Generally F is defined as a functor from $\mathcal{D}$ to $\mathcal{C}$, for ease of reading. Here we adhere our consistent usage of $F : \mathcal{C} \longrightarrow \mathcal{D}$.

[16] Functor compositions are different that arrow compositions. The arrow compositions are denoted as $f; g$ in this work, so as functor compositions $F; G$. However, when an object is of concern things need to change. For instance, an object, X of $\mathcal{C}$ under F is of $\mathcal{D}$ –that is $F(X)$. Then, this object can be transmitted by G into $\mathcal{C}$, then the corresponding object is $G(F(X))$. Thus, god is in the details.

> we have encountered, from free groups to limits to exponentials. [...] Indeed, I will make the admittedly provocative claim that adjointness is a concept of fundamental logical and mathematical importance that is not captured elsewhere in mathematics.

The introduction of adjoints[17] has expanded horizons in CT. The applications of CT realized that adjoint functors could define many fundamental mathematical notions (Awodey, 2006). For instance, firstly, consider the universal constructors in CT: every universal constructor can be defined in terms of adjoints.[18] All kinds of products, limits, and pullbacks, namely, all the fundamental constructions and their duals, can be described as adjoints (Peruzzi, 2006; Marquis, 2021). Furthermore, mathematical theorems can be described as adjoints, and even more, adjoints adhere to syntax and semantics: logical operators can be described as adjoints (Marquis, 2021). More precisely, adjoints can define propositional logical operations; and further, quantifiers as adjoints and introduction and elimination rules as the adjoint rules form a complete deduction system for quantificational logic (Awodey, 2006). But this is not the end of the story: adjoints also formulate a Heyting algebra and define its quantifiers! That is, adjoint inference rules formulize the intuitionistic propositional calculus (Awodey, 2006). That is to say, different logical models can be defined in terms of adjoints.

A second craft that adjoints bestow is that every free construction can be described as adjoints; in other words, free objects are characterized by adjoints (Peruzzi, 2006). A construction of free functor, a widely used free construction, can be an example here. Actually, the star of this example is a forgetful functor, denoted mostly by U. A forgetful functor is a functor that "forgets" the structure of the category that relates to the other category. That means U still preserves the structure of the first category yet forgets extra structures that the second category does not have.[19] For instance, there is a functor U from the category of complex vector space to the category of real vector space, such

[17] Although it is controversial, the introduction of adjoints is credited to Kan.

[18] We will not distinguish between left and right adjoints unless otherwise stated.

[19] One may use the notions of domain and codomain in order to specify the categories. Nevertheless, we believe this usage is set-theoretically loaded, which we avoid.

that $U : \texttt{Vector}_{\mathbb{C}} \longrightarrow \texttt{Vector}_{\mathbb{R}}$. U "forgets" any operation by i, since $\texttt{Vector}_{\mathbb{R}}$ knows nothing about imaginary numbers (Biss, 2003). Or, there is a functor U from the category of groups to the category of sets, such that $U : \texttt{Grp} \longrightarrow \texttt{Set}$. U forgets the group structure and the fact that arrows of `Grp` are group homomorphisms. The multiplication law is an extra structure in group theory that is not in set theory. The forgetful functor forgets this extra structure as well. That said, U is elementary; namely, it is trivial and non-informative. Said all, but what is the point of applying a forgetful functor and its relation with adjoints?

Adjoint functors can be taken as conceptual inverses (Marquis, 2021). In other words, a left adjoint can be considered the conceptual inverse of the associated right adjoint. As mentioned above, adjoints can describe universal properties; as such, there is a functor, which is a dual of forgetful functor, which can be described in terms of adjoints. Here comes *free functor*, conceptual inverse of forgetful functor.[20] Eventually, if a set A is given, there must be a way to define group structure in a non-isomorphic means, as there is a conceptual inverse of a forgetful functor from `Set` to `Grp`. Thus, this functor is to construct a group "freely" from the given set A. In other words, $F(A)$ is a group. And, $f : A \longrightarrow B$ of `Set` corresponds to the group homomorphism, $F(f) : F(X) \longrightarrow F(Y)$ of `Grp`. As, U and F are not isomorphic, rather they are conceptual inverses, for the given set A, $U(F(A))$ cannot yield the initial set A. On the other hand, there is a fundamental relation between A and $U(F(A))$: for an object, G of `Grp`, there is a morphism called the unit of the adjunction[21] $\eta : A \longrightarrow UF(A)$ such that given any function $g : A \longrightarrow U(G)$, there is a unique group homomorphism $h : F(A) \longrightarrow G$ such that $\eta; U(h) = g$. Moreover, $\xi : F(U(G)) \longrightarrow G$, called counit of the adjunction satisfy the following universal property: for any group homomorphism $g : F(A) \longrightarrow G$, there is a unique function $h : A \longrightarrow U(G)$ such that $F(h); \xi = F(U(G)); g$. Thus, h that is a group homomorphism is defined freely in terms of adjoints. Conse-

[20] A caveat is needed: one must be aware of the fact that `Grp` and `Set` are not isomorphic, or say, U cannot have an inverse functor, as it forgets some structure of groups, that do not persevere in `Set`. Forgetful functor has *conceptual inverse*, which is free functor.

[21] A pair of adjoint functors.

quently, adjoints determine an object or construction in terms of its relation to other given constructions. This is another way of saying that most of the constructions are just adjoints (Awodey, 2006).

The mentioned facts and the beauty of adjunctions can make them the focus of CT (Awodey, 2006), with their irresistible feature being "determination through universals" (Ellerman, 2007). Such a feature is not limited to mathematical theories: a third crucial aspect of adjoints is their ability to determine the structures of category theory within itself. In other words, the notion of adjunction has internally developed CT as giving rise to nontrivial questions about the theory in itself (Krömer, 2007). For instance, adjoints apply to provide conditions for the existence of limit in a category. Consequently, the notion of adjunction plays a central role in solving issues raised within CT, as well as the issues raised within mathematics, so that it helps CT develop itself.[22] Lastly and above all, this feature of CT can be named self-inspection.

In conclusion, the notion of adjunction is central in constructing a rigid theory of categories and investigating the other mathematical theories (Krömer, 2007). It applies every universal property to unify and subsume all the distinct universal properties (Awodey, 2006). In other words, adjoints can unify and/or subsume categorial constructions. Moreover, many mathematical theorems can be rewritten as statements about the existence of adjoint functors (Marquis, 2021). Hence, adjoints are the vital objects in CT.

4.4.8 Other Constructions

Many other constructions exist, such as cocones, pullbacks, and free monoids. In the following, formal definitions of the constructions will be provided whenever necessary -which is hardly- because giving categorial definitions of such constructions is of complexity even for a mathematics graduate. Then, for the

[22] Krömer (2007) interprets such development as an interplay between the development of the theory itself and the development of its applications. Nevertheless, the role of adjoints in developing both the theory and the practice is undeniable.

sake of simplicity and being within the limits of this work, from now on, we will only mention the names of the constructions, and we will focus on how these constructors are applied in several tasks.

4.5 Category Theory and Ontology 3.0

Employing category theory in knowledge representation is not a new idea. Healy (2010, p. 489) pronounces the value of category theory by presenting it as "a vehicle for the formalization of ontologies with mathematical rigor." As such, researchers focus on formalizing notions used in knowledge representations -such as in Entity-Relation (ER) systems or ontologies- with categorial constructions; for instance, a pushout over an alignment used for ontology merging and a natural transformation used for instantiation. The constructions,[23] such as a pullback, are not canonical; thus, they can be defined differently in each formalization. In other words, a pullback can be used for ontology merging in one setting, e.g., (Hitzler, Krotzsch, Ehrig, & Sure, 2005), in another, e.g., (Neuman & Nave, 2008), it can be used for concept formation. Furthermore, each framework has its own philosophical stance. For instance, philontological universals or set theoretical instantiation are defined in many category theoretical frameworks, which will be shown below; on the other hand, urtrope theory omits such things from the framework, which will be seen in the following part.

This part provides a limited number of researches on the employment of CT in knowledge representation, although the literature has myriad models for knowledge representation, ontology alignments, or constructions through CT. The works that we will not mention in this part can be found in the Bibliography, such as Colomb, Dampney, and Johnson (2001), Vickers (2010) or Schorlemmer and Kalfoglou (2005).

[23] Recall that in order not to overburden this work with technicalities, we are not giving categorial constructions in their full definitions; instead, we make do with their names and a fair amount of explanations.

Let us start with some business applications. Colomb et al. (2001) show that an enterprise model can be defined as an abstraction of the implementation model that instantiates it by employing cartesian morphisms and fibrations. The fact that the semantics of an enterprise model is closely related to the semantics of ER modeling exhibits an alignment between models. Another business application is from Wojtowicz (2016) who uses sketch theory[24] as a knowledge management framework for its ability to support modular development, uncertainty management, and dynamics in both metadata and instance data, to name some. He shows how to define models of a sketch, operate inferences on them, align different knowledge systems, and map between distinct knowledge representations in category theoretical terms.

Here are some examples of formalizing ontology generation and alignment[25] with CT. Kent (2010) takes ontologies as logical theories, and develops a metatheory based on category theory, called the *theory of institutions* for representing ontologies. In the theory, fibrations and indexed categories are used for representing logical theories, in other words, for building ontologies. Johnson and Rosebrugh (2010) focus on building ontologies in a category theoretic approach, creating new ontologies from the existing ones, and developing interoperating ontologies. Later, Johnson, Rosebrugh, and Wood (2012) develop their view and propose update processes as functors. Furthermore, Seremeti and Kougias (2013) target a network of ontologies whose components, viz., domain ontologies, interoperate and interact so that meaningful results can be extracted from aligned distinct ontologies. For this purpose, they initially represent an ontology as a path category and the category `Ont` of ontologies, in which the compositions of morphisms between ontologies render new ontologies, namely, a network of aligned ontologies.

Zimmermann, Krotzsch, Euzenat, and Hitzler (2006) propose a solution for merging disjoint ontologies by employing pushouts over alignments, which is

[24] A sketch is a graph-based KR that is actually a category together with some properties. More precisely, it consists of a directed graph together with a set of commutative diagrams in the graph.

[25] Ontology alignment, or ontology matching, is the process of determining conceptual equivalences between ontologies.

an attempt to meliorate ontologies of Ontology 3.0. Hitzler et al. (2005) follow the same line of thought and argue that the pushout constructions from category theory are a natural approach to the merging ontologies.

Healy and Caudell (2006) also define ontologies via category theoretical constructions. Their study of ontologies starts with the introduction of the category of `Concept`, whose objects are concepts, and morphisms are associations of the syntax of a concept T with a part of the syntax of a concept T, viz., $s : T \to T'$. The work continues by introducing an interconnected hierarchy of theories that unify knowledge-based systems. Moreover, Healy and Caudell (2006) formalize the incremental acquisition and representation of ontologies through adaptation in neural network architectures in category theory.

Probably one of the most famous ontology generation frameworks is offered by Spivak and Kent (2012) who introduce the *olog*, or *ontology log*, a category-theoretic model for knowledge representation. An olog is a category in which the objects are labeled phrases that refer to types of things, the arrows are, again, labeled phrases that refer to set-theoretical functional relations, and the commutative diagrams that are valid compositions refer to facts. For instance, the diagram in Figure 4.2 commutes, and the composition shows that a DNA sequence codes for a protein (Spivak, 2014, p. 34).

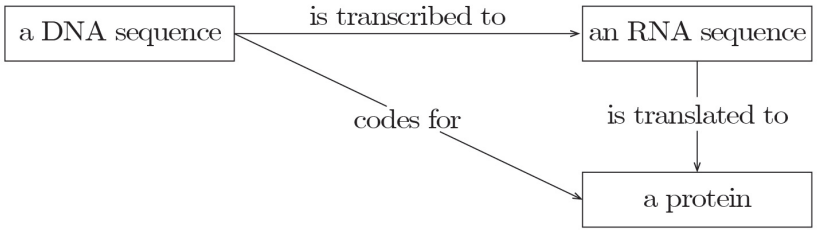


Figure 4.2: An example of a commuting diagram

On the other hand, not every as-if commuting diagram gives a fact. Consider Figure 4.3 from Spivak (2014, p. 35):

It seems legitimate to say that this diagram commutes, and the obtained fact is that every person lives in a city. However, it is false in many aspects. The most important aspect is that not everyone has to live in a city where their father lives. Thus, this is an example of a non-commuting olog.

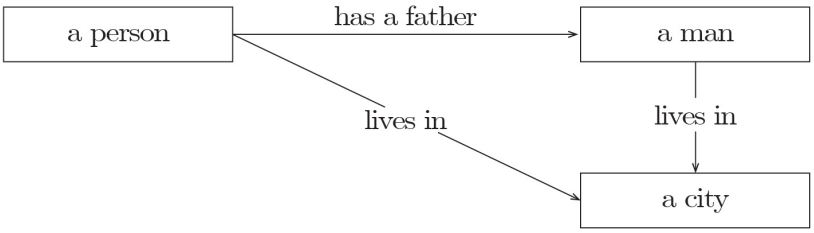


Figure 4.3: An example of a noncommuting diagram

Spivak and Kent (2012) also present an olog as a categorical database schema whose superiorities to a relational database schema are also shown. Later on, Spivak (2015) re-highlights a fundamental connection between categories and databases and develops a model based on CT that simplifies how we think about and use databases and that can solve classical database problems. As such, a schema is modeled as a category and data as a set-valued functor. Inspired by Spivak and Kent's work, Patterson (2017) introduces *relational ontology log*, or *relational olog*, that interpolates between ologs and description logic; by which a general recipe for constructing a categorical knowledgebase is proposed.

Lu (2005) proposes a typed category theory for representing knowledge. A typed category consists of a class of objects, a class of *typed* morphisms, along with type composition rules. He highlights that his typed categorial framework is representation independent; that is, the essence and properties of knowledge are category theoretical. Based on the work of Lu (2005), Wang and Rong (2007) introduce a knowledge model based on CT for the field of transferring emergency knowledge. They propose 'the knowledge pieces' that are micro-ontologies that are to be reorganized according to an emergency context. In other words, they aim to construct a category theoretical knowledge representation for the fields of emergency ontology construction and emergency knowledge reorganization.

Here are further employments of CT in KR. Bench-Capon and Malcolm (1999) work for developing an algebraic framework for specifying ontologies; and similarly, Cafezeiro, Haeusler, and Rademaker (2008) introduce a formal algebra for manipulating entities, ontologies, and contexts (See also Cafezeiro and Haeusler (2007) and Cafezeiro, Viterbo, Rademaker, Haeusler, and Endler (2014)). Their primary goal is to sustain the flexibility of the models; that is, the meaning

of entities can vary according to their context, and dynamic changes in the context can shape the indented use of the entities so that new forms of representing context are a requirement. For instance, from the formal framework, they propose that pullbacks and pushouts are the operations for entity and context integration, respectively. Besides, Coecke, Sadrzadeh, and Clark (2010) employ category theory along with vector space models in order to realize the proposition that when the machine can understand the meanings of individual words and grammatical rules, it can compute the meaning by using the principle of compositionality; that is, the meaning of a sentence rises from the meanings of its constituents that are combined with the grammatical rules.

Johnson and Rosebrugh (2010, p. 569) summarize an empirical observation as "over a wide range of practical studies finite limits and finite coproducts have sufficed to specify ontologies." That observation suggests that categorial constructions, even a few of them, are enough for knowledge representation tasks and ontology building, ontology alignment, and ontology merging in particular. However, all these applications are of Ontology 3.0. None provides an ontology that could pave the way for machine understanding, but this work. So, in the next chapter, we will answer the question of how the urtrope theory is to be implemented and explain the realization of the machine understandability in terms of urtrope and category theories.

4.6 Realization of the Urtrope Theory

We have introduced that Ontology 1.0, Ontology 2.0, and Ontology 3.0 mainly represent the world in terms of entities and relations in one way or another. This picture shows us that the main representational form of these ontologies is set theoretic: entities as concepts and functions as relations. To illustrate this form, let us examine the Ontology 3.0-style of ontology generation in particular.

Representing the world or a portion of it starts with figuring out what the entities are. So, the very first thing an ontologist does is list the entities: the

upper-level categories[26] in the case of upper-level ontologies or the entities in the context in the case of domain ontologies. Then comes the second task of figuring out the network of relations. Relations are determined according to the context. Since the semantic properties of entities are pre-determined, the boundaries set by the context cannot be exceeded, even if there can be some other relations between the listed entities. The second step of ontology generation is to choose a formal system for implementing the ontology in the machine. Mazur (2008, p. 6) states that a formal system representing a mathematical theory must have all its mechanics and vocabulary to argue and produce proofs. However, we have already examined that the formal systems mainly used in Ontology 3.0 are set theoretical in their nature and whose implementations are troublesome in many aspects. To solve the consisted puzzlements, we noted that another ontological stance that is particular to the machine must be taken and that another way of implementation is required to eliminate the problems of a set-theoretical approach. Consequently, upon our examination, urtrope theory is the best candidate for a machine ontology, and category theory fits very well to implement it. Accordingly, prioritizing relations over entities, Ontology 4.0 comes into existence in the machine via a mathematical theory whose mechanism and vocabulary argue and produce proofs in a distinctive, if not absolutely distinctive, way than Ontology 3.0 does. Additionally, Mazur (2008, p. 6) contrasts a categorical system with a set theoretical system by stating that the former says nothing about proofs but captures the essence of what a mathematical theory consists: of within the works with category theory, proofs just emerge simultaneously. That is a considerable distinction, indeed.

The roadmap in line with the aim of this thesis is as follows. Relations represent the world: everything, either an object or an event, must be defined and understood in terms of relations. To this end, we first discussed this issue ontologically and concluded that trope theory is the best to be adopted. On the other hand, we realized that we need *non-extensional* building blocks to empower the representational form. Thus, we proposed the *urtrope theory*, which is trusted to be

[26] The realist perspective is taken.

the best ontological perspective that paves the way for machine understandability. Then, we introduced *category theory*, which is also trusted to be the best formalization tool that vivifies the urtrope theory. As such, we purported that category theory applies the urtrope theory with some modifications. So, in this part, we will show that the urtrope theory can be realizable mathematically on the categorial foundations. If this is done successfully, we will legitimately name the marriage of the urtrope theory and category theory as *Ontology 4.0*.

4.6.1 Introduction: Categorial Constructions and Their Equivalences

Applications of category theory are not limited to mathematical theories. Various applications of CT are developed in logic, computer science, linguistics, and philosophy. Investigations in these fields, including ours, reveal or discover facts that are specific to the fields in light of the triple correspondence: category theory, intuitionistic logic, and lambda calculus; in other words, category theory, logic, and computing.[27]

A detailed correspondence is given in Table 4.1 by Peruzzi (2006, p. 448) (See also Awodey (2006)).

Table 4.1: Basic correspondence between logic and category theory

Basic correspondence	
Typed λ-calculus	Cartesian closed categories
Untyped λ-calculus	C-monoids -without terminal object-
Martin-Löf type theories	Locally cartesian closed categories
Intuitionistic (local) type theories	Toposes

This table guides us to realize the urtrope theory: the correspondence between untyped λ-calculus and C-monoids -without terminal object- can lead to a for-

[27] Here is another way of stating it as computational trinitarianism: categorial rules, Gentzen logical rules, and typed programming. Or, the triple is called Curry–Howard–Lambek correspondence that refers to the three-way isomorphism/analogy between intuitionistic logic, typed λ-calculus, and cartesian closed categories. According to which, propositions as types or as objects, and proofs as terms or as morphisms.

malization of urtropes. Thus, there should be an architecture that is founded on C-monoids -without terminal object- and erected through cartesian closed categories that gain new properties on each level of construction of Ontology 4.0, which is the realization of urtropes via category theory. Then, tropes are to be defined in terms of C-monoids –without terminal object-which are cartesian closed categories that provide calculations with types. Nevertheless, recall from typification discussions, Martin Löf's type theories were admitted for generating types of types. So, tropes must have some additional properties, and at the end of the day, they should be regarded as toposes. Let us examine these purported equivalences and representations, starting with urtropes.

4.6.2 Urtropes in CT

Our investigations on Data 4.0 showed the need for an untyped universe, on which the introduction of urtropes was hinged. This section shows how CT represents non-extensional ontological objects in an untyped fashion.

Table 4.1 shows the correspondence between C-monoids –without terminal object- and untyped λ-calculus, where urtrope formalization is based. Before diving into this correspondence and its consequences, let us introduce what a monoid and a C-monoid are and announce that from now on, we are not going to deal with formal definitions of these constructions, along with exponentials, (finite) limits, (pre)sheaves, evaluation maps, or sketches. Even though providing their definitions would enlighten our categorial understanding, their glare can make one, who has limited category theory knowledge, blind.

A monoid is a category with one object: $\mathcal{C}$ is a category of a monoid M, such that M is the only element of $\mathcal{C}$; all the morphisms are of the form $f : M \longrightarrow M$; the unit element of the monoid is the same as the identity arrow of $\mathcal{C}$; and for any f and g of $\mathcal{C}$, there is h, such that $f;g = h : M \longrightarrow M$ with the property that composition of arrows corresponds to multiplication of monoid elements.[28]

[28] That is composition is a binary operation: $f;g = f \times g$; LHS is an operation of $\mathcal{C}$, and RHS is an operation of the monoid.

A C-monoid is a monoid with extra features, such that $\mathcal{C}$ be a cartesian closed category, with an object M which is isomorphic to M^M and M, and not isomorphic to the terminal object (Scott, 2000; Lambek & Scott, 1984). Meanwhile, $M \cong M^M \cong M \times M$ are satisfied.[29]

Every monoid brings about a category. That is to say, an object with its identity arrow and arrows, if there are any, can form a category that satisfies the laws of associativity and composition. When an urtrope is identified as a C-monoid, an urtrope is a category whose relations with other urtropes make them and their arrows visible. Moreover, employing C-monoids realizes a typed space out of untyped representational units; indeed, types naturally emerge from untyped (Cardelli & Wegner, 1985). This brings us to the Curry view of typing.[30] More precisely, untyped terms come together to form typed terms by using type inference rules for assigning appropriate type schema (Scott, 2000).[31]

Without further ado, we have adopted the C-monoids as a categorial model of the urtrope theory, an untyped system.[32] Our adaption can be integrated with Hines (2016)'s theory of self-similarity where a C-monoid, say M, M satisfies $M \cong M \times M$. This approach can be utilized for illuminating urtropic constructions, as a category generated by a self-similar object or the category freely generated by a self-similar object allows to use the isomorphisms freely to play with the typing of arrows. That can pave the way for producing untyped analogs of many categorical properties.

We neither know nor care about the object of a C-monoid, so identifying an urtrope hinges on its functorial relations with other urtropes. Moreover, in light of the Curry view of typing, we claim that the types are determined by the patterns of arrows between urtropes. Urtropes by themselves cannot determine a type, but a particular morphism composition between urtropes emerges a type;

[29] For the historical introduction of C-monoids see Hines (2013).

[30] As opposed to Church, who suggests starting with "typing" the untyped terms.

[31] For details, search on the categorial combinators.

[32] For a detailed investigation of C-monoids, untyped systems, and untyped λ-calculus, refer to Lambek and Scott (1988), pp. 101ff.

that will be investigated in the following part. Lastly, it is worth mentioning that the crux of this matter is that both urtrope theory and an untyped system take everything on the same level: urtropes are the building blocks of everything, and untyped monoids are the building blocks of a type system. Now, let us explore tropes in CT.

4.6.3 Tropes in CT

From the background of the urtrope theory, we know that tropes, the primitive ones, are compositions of urtropes, and the complex ones are compositions of urtropes and primitive tropes, or primitive tropes. They, machine-ontologically, are relations out of which every philontological and whatever object is constructed. As such, urtropes are the non-extended, whereas tropes are the most primitive extended building blocks of reality.

Let us illustrate a trope in categorial language and start with defining a primitive trope.

Definition 17 *A primitive trope is a category consists of*

1. *A collection of urtropes,*

2. *A collection of types,*

3. *A collection of typed arrows between urtropes, where a typed arrow f is denoted as $f : X \xrightarrow{t} Y$, where X and Y designate urtropes, and t is the type of f,*

4. *For every urtrope X, there exists an identity arrow, id_X,*

5. *There is a unit type, u, such that for any urtrope X, $Mor(X, X, u) = id_X : X \xrightarrow{u} X$,*

6. *For types t and s, there is a type $w = t \times s$ that is the composition of types t and s,*[33]

7. *The composition of arrows is defined as follows. Let f and g be arrows with types t and s, respectively. If $f : X \xrightarrow{t} Y$ and $g : Y \xrightarrow{s} Z$, then $h = f;g : X \xrightarrow{w} Z$,*
   Associativity and identity laws hold.

A complex trope is a trope that contains at least one primitive trope. Here is its definition:

Definition 18 *A complex trope is a category consists of*

1. *A collection of urtropes and primitive tropes,*

2. *A collection of types,*

3. *A collection of typed arrows between urtropes and primitive tropes, where a typed arrow f is denoted as $f : X \xrightarrow{t} Y$, where X and Y designate objects, and t is the type of f,*

4. *For every object X, there exists an identity arrow, id_X,*

5. *There is a unit type, u, such that for any object X, $Mor(X, X, u) = id_X : X \xrightarrow{u} X$,*

6. *For types t and s, there is a type w in the collection of types, such that $w = t \times s$ that is the composition of types t and s,*[34]

7. *The composition of arrows is defined as follows. Let f and g be arrows with types t and s, respectively. If $f : X \xrightarrow{t} Y$ and $g : Y \xrightarrow{s} Z$, then $h = f;g : X \xrightarrow{w} Z$,*
   Associativity and identity laws hold.

[33] The collection of types is closed under composition. For the sake of simplicity we omit the details of composing rules.

[34] The collection of types is closed under composition. For the sake of simplicity we omit the details of composing rules.

Let us start with a caveat. Urtropes and primitive tropes are categories *per se*; thus, there are two different kinds of morphisms between urtropes and primitive tropes. Either in the definition of primitive or complex tropes, the objects – urtropes and primitive tropes– are categories, and the morphisms are arrows. On the other hand, they are taken as categories outside a trope category. The morphisms are functors, which examine the structural relations between these categories, and whose nature and function are thoroughly different from arrows. Once again, a morphism in a category, namely an arrow, tells how the objects are connected to each other. In contrast, a morphism between categories, namely a functor, tells things about the structures of categories. Therefore, primitive tropes as objects are connected to each other precisely in a complex trope, and any investigation between primitive tropes through functors is another story. Consequently, urtropes or primitive tropes as objects are processed without penetration, namely, without tearing them apart or interrupting them. As far as the definition of complex category is satisfied, no concern would be left.

Looking at what the tropes will be represented in the CT by looking at Table 4.1, we can say that they are cartesian closed categories. More precisely, as we defined tropes as typed categories and there is a correspondence between typed λ-calculus and cartesian closed categories, a trope can be represented by a cartesian closed category (CCC). Indeed, C-monoids, which are the correspondence of urtropes, are also founded on CCCs. So, the following definition lies at the heart of the formalization of the urtrope theory.

Definition 19 *A cartesian closed category, $\mathcal{C}$, is a cartesian category, i.e. a category with all distinguished finite products, for all whose objects, X,the functor $(\breve{}\breve{}) \times X : \mathcal{C} \longrightarrow \mathcal{C}$ has a specified right adjoint. Thus there is an isomorphism for any Y and Z of $\mathcal{C}$, $Mor_{\mathcal{C}}(Z \times X, Y) \cong Mor_{\mathcal{C}}(Z, Y^X)$.*[35]

[35] See Scott (2000) for details.

A CCC is a category of map objects/mapping objects/exponential objects denoted with Y^X. Such objects are important because they are used for constructing objects satisfying a universal property in categories.

Although a CCC is the main category to be used in the formalization of the urtrope theory, from Table 4.1, one can derive that if type theories are indispensable for the urtrope construction to occur, as it is, representation by toposes is required. In other words, it is seen that a more specific formalization of tropes is required than the representation of the CCCs, which is of toposes.

A *topos*, singular form of *topoi* or *toposes*, is a cartesian closed category with an extra structure that produces subobjects for each objects. More precisely,

Definition 20 *A topos is a category that has all finite limits and all exponentials with a subobject classifier.*[36]

4.6.3.1 What is a Topos?

The notion of 'topos' was introduced by Alexander Grothendieck, whose aim was to "provide a mathematical underpinning for the 'exotic' cohomology theories needed in algebraic geometry" (Caramello, n.d.). The works of Grothendieck suggest that a topos can be regarded as a *generalized space*, where categories of sheaves of sets can replace topological spaces.[37] Based on the studies of Grothendieck, William Lawvere and Myles Tierney transform the notion of 'topos' into a *mathematical universe* (ibid.). That is, each topos can represent its own mathematical framework. These two developments in category theory cultivate the topos theory, where different mathematical theories can be systematically studied through different settings of toposes. So, topos theoretical

[36] Alexander Grothendieck invented the concept of 'topos' during his studies on algebraic geometry. However, the notion of 'topos' we are dealing with comes from Lawvere and Tierney's work. In order to differentiate these distinct notions, the former is called a 'Grothendieck topos,' and the latter is called an 'elementary topos' (Baez, 2021). On the other hand, every Grothendieck topos is an elementary topos (Caramello, n.d.).

[37] Although the notion of 'sheave' is not complicated to explain, its explanation is too long and not essential for this work.

techniques allow finding out deep connections through which a context can be studied in a different context. Furthermore, they also allow comparing different mathematical theories or objects. Such comparisons help find out relations between such theories or objects.

Bridging representations Thanks to the notion of Grothendieck toposes, different mathematical theories can be related to each other, and it is possible to study them from various points of view: A topos can behave as a bridge that connects different theories (Caramello, n.d.). This is possible by representing different theories, which can be in different languages, are isomorphic to a topos. That is to say, a topos provides a framework in which relations between different theories can be expounded.

Theoretically, let A and B be different theories, and $A^{'}$ and $B^{'}$ represent the classifying toposes of A and B, respectively, which are constructed by characterizing them for the same invariants. $\mathcal{D}$ is a common classifying topos when $\mathcal{D}$ is equivalent to both $A^{'}$ and $B^{'}$. So, $\mathcal{D}$ can be used as a 'bridge' for transferring knowledge between A and B (Caramello, 2018, p. 69). Figure 4.4 illustrates this feature of toposes.[38]

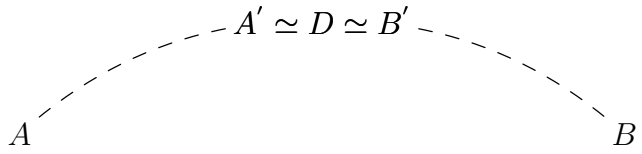


Figure 4.4: $\mathcal{D}$ as a briding topos between theories A and B

Toposes and Logic The typed λ-calculus is the internal language of CCCs, so toposes have an internal type theory in the λ-calculus (Patterson, 2017). That means, indeed, any logical theory can be understood in a topos. Ω, the subobject classifier, provides different degrees of truth-values to be defined in a topos.[39] It

[38] Theoretically, $A^{'}$ and $B^{'}$ are equivalent, so there is no need for a topos, $\mathcal{D}$ when the same invariants characterize both theories.

[39] But what is a subobject? As we mentioned, CT generalizes mathematical ideas, and the idea of the subset can be generalized in topos theory by introducing subobjects; moreover, it is a generalization of parts, pieces, and inclusion. Inclusion is defined differently in different theories; CT abstracts

relates certain arrows of a category with notions of truth in the same category. As notions of truth are changeable from context to context, truth-values of a system are not limited to a two-valued system of classical logic. For instance, in category of sets truth-value object is defined as $\Omega = \{0, 1\}$; in category of graphs there are intermediate truths, where 0 is $True$, ∞ is $False$, and the rest have gradual degrees. In categories representing a dynamic system, some objects in the subdynamical system will be true after some update, yet some discrete objects cannot be in the subdynamical system; they can never be true no matter how many updates occur (Lawvere & Schanuel, 2009). So, in a sense, toposes can be interpreted as categories with an internal system of truth. Various logical systems with different degrees of truth, even the dynamical ones or informal logics, can be represented in topos theory (Vickers, 2010). Thus, toposes provide the bedrock for various sorts of logics. Translations between logics, comparing or bridging them, and transforming one logical style into a collection of styles are possible (Cf. Vickers, 2010). For instance, a problem in a category having geometric logic as its internal logic can be solved in another category having first-order logic as its internal logic; or vice-versa, one can manipulate the former category and do non-classical logic based on the reasoning in geometric logic, even without admitting double negation. Consequently, toposes are an algebraic encoding of deductive systems, and various categories correspond to well-defined logical systems with their deductive systems and completeness theorems (Marquis, 2021).[40]

In addition to these, logical operators and quantifiers can also be modeled by morphisms. That is to say, the alleged primitive operations and quantifiers in different sorts of logic are constructed using a truth-value object, and universal constructors (Awodey, 2006). Having said that, set theoretical concepts, such as intersection and union, can also be translated into the categorial language

and defines it as a subobject. Subobjects are defined through monomorphism. Moreover, CT can also define arrows *in* arrows: x is in α, if $x = \alpha \circ k$ where $x : T \longrightarrow X$, $\alpha : A \longrightarrow X$ there is unique k such that $k : T \longrightarrow A$.

[40] For a list of categorical logic see Marquis (2021). This proves that regardless of the philosophical view of mathematics, categorial tools in logic can model any logical system. Above that, there can be translations between those logical systems thanks to their categorial settings.

thanks to Ω (Asperti & Longo, 1991).[41] Furthermore, objects in a topos can be described as if they were sets so that statements can be proved in a topos. Anything in a logical theory can be represented in a topos theoretical setting. However, in logical theories –those based on set theories are not exceptional– ontological commitment is an issue: operators, quantifiers, and rules are interpreted within some philosophical stances. Since topos theory is an abstraction, it can provide "a neutral common frame for the formalist, intuitionist, Platonist perspectives, because differences in *ontological commitment*" (Peruzzi, 2006, p. 450). For instance, a finitist can build the topos of finite sets and functions between those to work with only finite sets; a physicist can build the "smooth topos" to do calculus with infinitesimals; or a constructivist can build the "effective topos" to work with "effectively constructible" sets and "effectively computable" functions: great numbers of toposes can be built and hand-crafted to meet specific needs (Baez, 2021).

The Computational Trinitarianism The statement of Peruzzi (2006, p. 432), "[a]s the 21st century opens, it is of paramount importance to indicate a direction along which mathematicians and philosophers can collaborate so as to regenerate trust in understanding, objectivity and explanation," has a missing component: computation. In other words, we believe that the twenty-first century demands the collaboration of philosophers, mathematicians, and computer, information/data scientists to "regenerate trust in understanding, objectivity and explanation." Moreover, we also believe that category theory is the study of the century: the representation language is the formal representation that can also be used as an implementation language. Therefore, choosing CT as a bedrock of representation and computation solves three problems with one solution (Cf. Wojtowicz, 2013).

[41] For this reason, toposes are considered more fundamental than ST, since simpler structures can express its primitives. For instance, conjunction of propositions in a deductive system is an instance of a product (Marquis, 2021). Of course, standing alone, this cannot be the only reason for replacing ST with CT, in particular topos theory, or even acknowledging it as *the* foundations of mathematics.

Category theory is known to provide not only an ontological neutral frame but also a self-contained and unified approach to information systems. Lambek's proposal of the category of graphs[42] is an example of a deductive system (Cf. Lambek, 1980). The components of a deductive system are formulas, deductions/proofs, and the rules of inference, and their correspondences in category theoretical notions respectively are objects, arrows, and the operations on the arrows (Marquis, 2021). As mentioned before, a topos can be taken as an algebraic encoding of a deductive system, which is at the heart of computation. In this part, then, we will explicitly account for the inseparable the computational trinitarianism: categorial rules, Gentzen logical rules, and typed programming, which is based on the basic correspondence shown in Table 4.2.

Table 4.2: Basic correspondence between logic, category theory, and proof theory

Basic correspondence		
Tayped λ-calculus	Cartesian closed categories	Intuitionistic proof theory

In Gentzen's approach, namely in natural deductions, logical connectives get their meanings from how they are used in inference. That is to say, in order to understand the meaning of a connective, one must understand the inference rules that govern its use. For instance, grasping the meaning of $\wedge$, *and*, requires grasping the rule that from $X \wedge Y$ one can derive X.[43] In this respect, the focus is shifted from the study of truth to *the study of inference*. A topos, the internal formal language of a category, presents a natural deduction system. For instance, both in λ-calculus and combinatory logic, functions are taken as computational processes that take an argument and return a value; so, in this sense, functions are rules. In this vein and based on basic correspondences, morphisms are rules, and rules of natural deduction, e.g., elimination rule, on morphisms and their compositions —along with the fact that a natural deduction system is a decidable and syntactically decidable proof system, which makes it be called an automated

[42] A category whose objects are arrows and dots, and whose morphisms are arrows between them, as $s : Arrows \longrightarrow Dots$ and $t : Arrows \longrightarrow Dots$, where 's' for the source and where 't' for the target.

[43] That is $\frac{X \wedge Y}{X}$.

proof system– is the inference system of Ontology 4.0. Thus, Ontology 4.0 is a system whose inference system is within its representation. This is another saying that its syntax and semantics are the same things.

There is no getting away from the untyped aspect of computation. Scott (2000, p. 24) highlights this by uttering, "there is an underlying untyped aspect of computation, already going back to the original work on lambda calculus and combinatory logic in the 1930s, which often underlies concrete machine implementations." This fact is the base of Ontology 4.0, where urtropes can be represented as C-monoids that are cartesian closed categories. That is to say, a typed system springing from an untyped system is tenable in a urtrope theoretical framework. The close connections of cartesian closed categories to typed λ-calculus admit theoretical foundations of Ontology 4.0; indeed, and above all, the close connections of CCCs to intuitionistic proof theory admit practical, namely implementation foundations of Ontology 4.0. Just as the Curry-Howard-Lambek isomorphism is "the cornerstone of modern programming language semantics" (Scott, 2000), implementation of the equivalence between these three notions is the key to the *realization of the machine-understandability*.

Consequently, CT guarantees that all the mathematical theorems can be defined as categories; thus, any mathematical model of a representation can be translated into a category. Thanks to topos theory, two different mathematical theories can be studied, compared, and analyzed by constructing a common classifying topos. Tshe concept of *subobject classifier*, or sometimes called as *truth-value object*, Ω, is significant, as it is a great tool for discovering and constructing logical aspects of toposes. Hence, topos theoretical methods allow figuring out deep connections between mathematical theories, and via those connections, knowledge transfers are possible (Caramello, n.d.).

4.6.3.2 Tropes as Toposes

At the end of all this, we have seen what a topos is. Its unique place in our work primarily represents the tropes: urtropes are C-monoids, and tropes are toposes. There is only one caveat here. Just as we do not take the theory of types as it is,

we cannot take the topos theory as it is. The topos theory, within the framework of this study, must always be based on the urtrope theory. Therefore, objects in topos theory must ultimately be rendered as C-monoids.[44] So, our primary purpose of using topos theory is to represent tropes.

The Category `Type` Cardelli and Wegner (1985, pp. 483–484) highlight that using a type system inhabits relations between types:

> The usefulness of a type system lies not only in the set of types that can be represented but also in the kinds of relationships among types that can be expressed. The ability to express relations among types involves some ability to perform computations on types to determine whether they satisfy the desired relationship. Such computations could, in principle, be as powerful as computations performable on values. We are concerned, however, only with simple, easily computable relationships that express uniform behavior shared by collections of types.

We use and interpret this explanation to elaborate on how type-composition rules are determined. Type composition rules must be defined in a categorial framework, for which the category `Type` of types can be proposed. The objects of `Type` are types, and the arrows of `Type` are typing rules. Thus, arrow compositions are type composition rules. As types specify the constraints, namely the consistent interactions, `Type` would specify the allowable morphisms. In other words, `Type` would specify the constraints on compositions.

This category is a perfect tool for constructing new types, which are, by definition, objects of `Type` that must satisfy all the laws of the definition of a category. For instance, assume that most of the tropes of a virus and a living system are known. `Type` can provide a list of semantic relations between them, and it helps to check whether a relation is allowable between these entities in either direction, namely from virus to living system and vice versa. Thus, the machine can automatically decide a network's possible interactions and derivations without other hierarchies or axioms. In sum, type constructions provide new types of occurrences and satisfy the axioms and laws pronounced in the language of CT. This tool, `Type`, will be used throughout Ontology 4.0 in different levels

[44] Our purpose in this study is not to show how to do this, which will be the subject of future work.

of abstractions and for different tasks, for instance, in type checking and type inference.

Starting with an untyped universe provides the opportunity of constructing the universe in different ways for different purposes since "[t]ypes arise naturally [...] from untyped universes" (Cardelli & Wegner, 1985, p. 473). Recall that different usage or behavior of an entity arises from the context at hand, and the context is determined through the relations between the entities it involves. Thus, from the urtrope theoretical perspective, arrow types determine the structure of a trope: classification of arrows and their compositions results in a well-defined type system.[45] Consequently, we can say that this system is based on the topos theoretical perspective, and `Type` is used where there is typification.

We have seen the relationship of toposes, which represent tropes, with type. So, what we call the structure of a trope will be the structure of a topos. In this case, we can say that we are talking about the types of topos. This is how we integrate topos theory into urtrope theory: we have found a way to make topos theory, type theory, and urtrope theory work together. Consequently, this whole system should be based on the topos theory. That is to say, one of the very fundamental components of Ontology 4.0 is the category of types, which is supposed to be grounded on the topos theory.

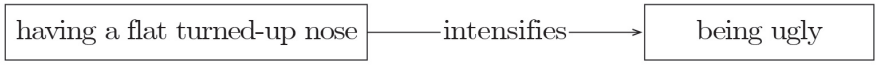
4.6.4 Entities in CT

In the urtrope theory, anything extensional consists of the most primitive *extensional* building blocks, viz., tropes, which consist of urtropes that are the non-extensional building blocks of all reality. The term 'entity' refers to anything in the world; a quark, the beauty of a line of code, an irrational number,

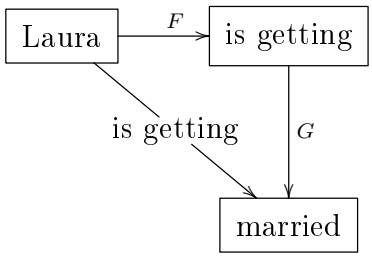
[45] However, one can raise "How are the arrows typed?" or "How can the collection of types of arrows determined?" Well, we leave these questions for the time being and will attempt to solve the problem of determining the types of arrows in the category of a trope with the help of the Yoneda Lemma in a future study.
Simply, the Yoneda Lemma says that a category can be reconstructed in terms of functors to sets, of which objects and arrows alike. That means reconstructing a category in terms of networks of relations is dealing with the entire category at once (Mazur, 2008). Moreover, it implies that an object can be studied basically by analysing *the* functor that it represents (Biss, 2003).

swinging, a yellowish star, Uzun İhsan Efendi, to solve the issue of gender discrimination in developed countries, and so on. In addition, (1) entities are trope compositions, and (2) we purport that they can be represented as toposes. Are, then, tropes[46] to be considered as objects, arrows, or both?

Recall that we spoke of "strategy of relativization," which refers to the idea that a methodological framework depends on the maxim of relativity, which says that the distinction between arrow and object is a matter of perspective (Krömer, 2007). Thus, we admit that an entity is a topos whose objects and arrows are both tropes. For instance, Soctares was known to have a flat turned-up nose and bulging eyes; he was notoriously ugly. Based on this information, some of his semantic properties are 'having a flat turned-up nose' and 'being ugly.' Now, consider the following diagram.



Here, the arrow $intensify$ has a certain type in the trope composition of Socrates. That also means that all these objects and arrows are urtrope compositions. Similar to the above, consider the diagram below.



This diagram is a context,[47] (we will see soon) where both ⌜LAURA⌝ and ⌜IS GETTING⌝ are objects, and *is getting* is an arrow.[48] Tropes that constitute ⌜LAURA⌝ or ⌜IS GETTING⌝ are connected to each other via tropes. Unlike set theories, to which we are accustomed, the following should draw our attention at this level: in a philontological parlance, properties and relations between

[46] 'Trope' is a generic term for referring to primitive or complex tropes unless otherwise stated.

[47] Contexts are represented as categories as well.

[48] The ⌜ ⌝ notation is used to denote the corresponding entity is taken as an category. How it is used will be explained soon.

properties are tropes. It is essential to get that things other than urtropes are toposes and their objects and arrows are either primitive or complex tropes. What distinguishes the role of a trope is its type. For this reason, a trope can be considered a topos, an object, or a morphism. Although this type of representation may sound counterintuitive, it gives us evidence of the richness of the representational system, viz., CT. And finally, it is helpful to remember that this ontology is not created for humans but for machine-understandability. Only this kind of ontology can make machines autonomous.

Representing urtropes as C-monoids and tropes as toposes is evident from Table 4.1, from which we cannot tell anything for entities. We chose to represent entities as toposes better to represent them by their internal structure and external relations. Because it is more reasonable for the machine to revise the semantic properties for the associations in different contexts instead of knowing an entity as a whole, this is why all the structures we represent as toposes, with the exception of urtropes, must be ready for structural change. It is only a topos theoretical approach that will allow us to make a *direct* analysis/comparison between tropes in different entities and entities in different contexts. Hence, the transfer of knowledge between two entities (resp. contexts) with respect to a trope (resp. entity) can straightly take place.

The bridging technique is not the only thing topos theory bestows to us. The existence of the exponential object will allow us to know more about the tropes within an entity. Because the condition of being able to know an entity, that is, to identify its trope compositions, will be a cumulation of crumbs of information coming from its relations with other entities. Let us unpack this with an example.

Consider the tropes in an individual, say, Socrates. ⌜SOCRATES⌝ denotes that Socrates is an entity category whose inner structure is under examination. We have chosen this representation instead of `Socrates`, which refers to the intext categorial representation of Socrates; in order to differentiate that the latter denotes 'Socrates' as an object or a morphism of another category. Recall that an object is studied in a category without penetration. Knowledge of the inner

structure of the object, comes from $Hom(Socrates, -)$ and $Hom(-, Socrates)$, end even from $Het(Socrates, -)$ and $Het(-, Socrates)$.[49] In what follows, such knowledge can reshape ⌜SOCRATES⌝. At this juncture, it is worth reminding that an entity category must be considered a construction that will expand. For this reason, entities, and even contexts, should be taken as pseudo-categories as Lu (2005) does. That will expand the machine's knowledge of semantic properties and entities.

On the other hand, since an entity is a trope composition, we can obtain information about its relationship with other entities in a context by using the information from its internal structure. At this very point, the exponential objects will provide information on all trope compositions in the context. So, the topos of ⌜SOCRATES⌝ will also give information about the types of $Hom(Socrates, -)$, $Hom(-, Socrates)$, $Het(Socrates, -)$, and $Het(-, Socrates)$.[50]

4.6.4.1 A Hierarchy of Semantic Properties

An entity includes many roles/behaviors, which are compositions of arrows, determined by trope compositions. A role/behavior is a composition of arrows. As the arrows are typed at the categorial level, then a role/behavior requires a categorial level investigation of arrows. We use this fact to determine a hierarchy of semantic properties. For instance, ⌜SOCRATES⌝ includes a man-trope.[51] The trope composition of ⌜MAN⌝, which necessarily contains trope composition of ⌜HUMAN⌝; moreover, it contains trope composition of ⌜LIVING BEING⌝. That is to say, trope compositions, namely the types, will eventually give a hierarchy of semantic properties. Unlike the previously mentioned frameworks, where a hierarchy must be mentioned manually, each type necessarily exhibits its sub-

[49] Heteromorphims refer to morphisms between two objects of different categories. This kind of morphism will be investigated in sectionAutonomy/Active agency

[50] The Yoneda Lemma is a powerful tool for examining such knowledge transfers.

[51] When this trope is not in the collection of arrows of the category of Socrates, it is because either there is missing information; viz., some arrows or the typing rules that cannot allow a trope composition with a man-trope in the case of ⌜SOCRATES⌝, or something that we have not recognized so far.

types via the patterns of the trope compositions. Accordingly, and consequently, background knowledge is already there.

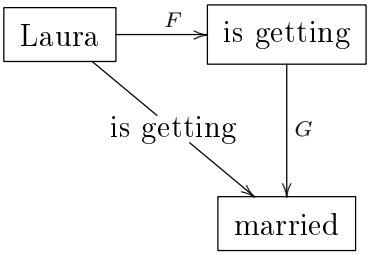
4.6.4.2 Possible Semantic Properties

Possible semantic properties of an entity are also expressible in its trope composition. For instance, a man is potentially a husband, and from the other way around, a husband must be a man –of course, in the scope of the old-fashion classification of the institute of marriage. A trope composition of ⌜HUSBAND⌝ includes a trope composition of ⌜MAN⌝; and further, a trope composition of ⌜HUSBAND⌝ is *potentially* included in a trope composition of ⌜MAN⌝. This is reasonable since every trope composition consists of two parts: the core is the foundational trope composition that ever changes, and the peripheral is the contingent trope composition. As such, the core and a part of the peripheral of ⌜MAN⌝ are included in the core of ⌜HUSBAND⌝, whereas the peripheral of ⌜MAN⌝ contains some arrows and their compositions that end up with a husband-trope.[52] That said, it is not necessary to know that the structures of one entity are inherited by another in advance. Since they are both categories, a functor between them can reveal such properties.

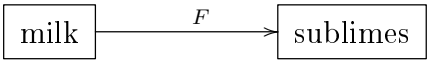
4.6.4.3 Constraints on Entities

Constraints on entities are also expressible in trope compositions. Let us illustrate detectable constraints on individuals. Suppose that one wants to know whether Laura is getting married is possible. Let us use the previously given context:

[52] Thus, a husband-trope is more complex than a man-trope.



F, G, and *is getting* are arrows, and `Laura`, `is getting`, and `married` are objects. The existence of $F;G$, namely *is getting*, depends on types of F and G,. However, for this diagram to commute, there need to be additional conditions, such as Laura being older than 18 to get married. Thanks to the decomposition facility of the urtrope theory, every object and arrow can be represented at the urtrope level so that whether the arrow composition is allowable or not can be figured out. In other words, philontological interactions between sources, targets, and relations can be examined either at the trope level or at the urtrope level, namely at the same existential level. This example illustrates not only that the possible sources and targets can be matched with the relation `is getting`, of some type, but also that *is getting* emerges when the constraint of Laura's age is satisfied in the context. That is, if ⌜LAURA⌝ has no trope of older-than-18, then there is no functor of $F;G$ from ⌜LAURA⌝ to ⌜MARRIED⌝. Furthermore, it is detectable that a morphism links two entities. For instance, given that sublimation refers to the process of changing from a solid state to a gas state without passing through a liquid state, it is a fact that milk cannot sublime.



Thus, there is no F at the trope level examination since there are no possible trope patterns that allow such a morphism between the categories. Moreover, a trope-level examination denies the existence of a context category, "my cat sublimes milk," as the context entities cannot have allowable arrows.

So, there are no F and G since there are no possible trope patterns that allow such transitions between the categories. It cannot be detected that the pattern

of change-directly-from-the-solid-to-the-vapor-state neither in ⌜MILK⌝ nor ⌜MY CAT⌝, even though it can be detected so many possible morphisms that provide $F;G$; *spills*, *hates*, *steals*, or *drinks* to name some.

The last two examples examine entities in propositions since they are only known within interactions. Based on these examples, one can say that entities with the same type have the same trope composition internally, and when they are used as objects externally, they behave similarly. That is to say, the types of entities depend on their internal and external relations: the internal relations are gained through trope compositions, and external relations are gained through the departing and arriving arrows in a category.

4.6.5 Contexts in CT

Let us begin with a declaration: for the sake of simplicity, we delimit our investigation to propositions and facts, yet other types of sentences can be studied in a machine ontology; a representation and its inference system are one and the same thing in CT, thanks to Curry-Howard-Lambek correspondence. For instance, an interrogative system in CT that deals with propositions and questions includes its composition rules *per se*.[53]

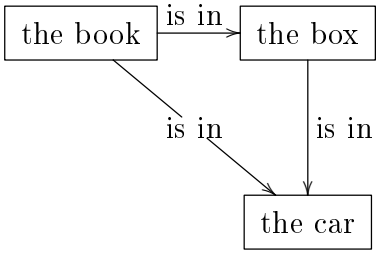
A context consists of at least one proposition. An entity alone, such as 'Laura,' means nothing, just like 'think' has no meaning at all by itself. On the other hand, "Laura is thinking" expresses a sense. That said, a context contains the source and target entities and relations that are precisely connected to each other. For instance, "a vase is typing" has no sense. If we want to give the structure of this *precise way*, then we can construct a category for a context. A context as a category consists of entities as objects and relations as arrows.[54] Moreover, as explicitly declared, a context is represented by a topos. In its

[53] This issue of composition rules will be visited at the end of this part.

[54] Compared with the other frameworks, the objects-arrows are mathematical abstractions of OWL's classes-properties, Sowa's concepts-conceptual relations, Quillian's nodes-links, Chen's entities-relationships, and Allen's events-time interval relations (Lu, 2005).

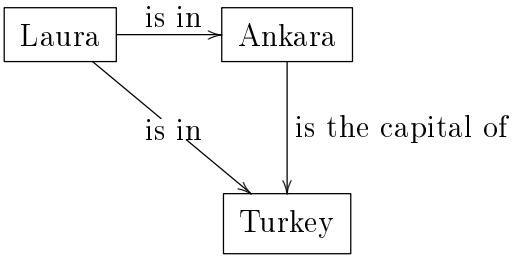
simplistic explanation, a proposition is a structure in the form of source-relation-target, where in some cases, the source and target can denote the same thing. In machine ontology parlance, a proposition is an arrow in a context category.[55]

Suppose we are given a context consisting of two propositions: "The book is in the box" and "The box is in the car." Using only these two propositions, one can reach a conclusion that "The book is in the car." This inference can be shown as a commuting diagram:



Therefore, arrow compositions in this category are inferences. Alternatively, Spivak (2014) uses the term 'fact' instead of 'inference' and defines it as "a commuting diagram."

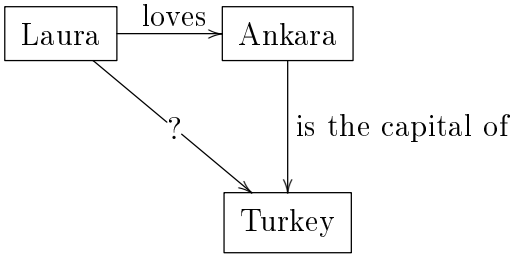
Type compositions are structured in `Type` by the rules $f : X \xrightarrow{t} Y$ and $g : Y \xrightarrow{s} Z$, then $f;g = h : X \xrightarrow{t \times s} Z$. `Type` includes $(is\ in$:physical);$(is\ in$: physical) = $(is\ in$: physical), where "physical" is a type of $is\ in$ relation, as in the above example.
Consider another example:



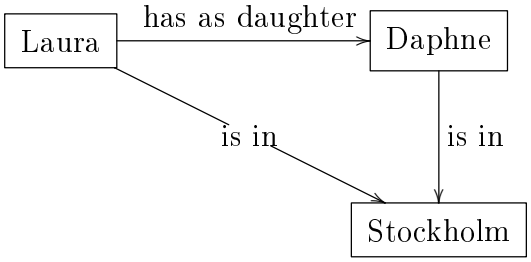
This diagram commutes since the typing rules allow the transition through

[55] Laura —eats→ apple is a proposition, so as Laura —eats→ □, since the blank category, which is indefinite pronouns, is determined by the functor. So, here the blank category is ⌜SOMETHING⌝.

(*is in*:physical) and (*is the capital of*: physical), and the transition is (*is in*:physical). Of course, the trope compositions of ⌜LAURA⌝, ⌜ANKARA⌝, and ⌜TURKEY⌝ have roles in figuring out the types of the morphisms. Nevertheless, there would not always be commutative diagrams, or say, there would not always occur facts. Lu (2005, p. 755) provides two situations. The first one says the type product $t \times s$ does not exist necessarily, as in the following example.



The second one says that even if the type product $t \times s$ exists, there would be no such h necessarily. See the following example.



has as daughter;*is in* may compose to *is in*, but not necessarily. In such a scenario, the composition morphism cannot be determined by typing rules, but when all the entities are examined at the trope level or some other entities come to the scene, whether Laura is in Stockholm can be figured out.

In conclusion, an entity, as an object, is in the proposition for one of its aspects. That is, each aspect of an entity is privileged in some propositions and not in others. Moreover, some aspects of an entity are privileged in some contexts and not in others, for a context may include several aspects of an entity. The semantic properties of an object can be represented as a quotient/slide/partial category, whichever is the most appropriate, and the machine can figure out the relation type with adjoints and/or some constructions. It is the context that glues semantic properties of entities and that expels entities that have no

appropriate semantic properties, and it is the typing rules that make this whole idea work.

4.6.5.1 Typed Objects in Contexts

Consider the following context: "The book is in the movie" and "The movie is in the festival." These propositions cannot yield a fact, for it may be either that there is no such type product or that even if there is such a type product but no such arrow for this context. However, such situations are not problematic, as Lu (2005, p. 755) offers a pseudo-category, an algebraic structure similar to the typed categories with some modifications.

In the above example, the reason a fact did not occur was the output from arrow compositions. Nevertheless, one can reach the same conclusion from the types of objects. As such, the type of 'movie' in the first proposition is different from 'movie' in the second one, so there is no transition through the object 'movie.' However, the objects are not *typed* in CT. Recall what Mazur (2008, p. 19) states "an object X of a category $\mathcal{C}$ is determined by the network of *relations* that the object X has with all the other objects of $\mathcal{C}$" [emphasis added]. From another perspective, as we have seen in the examples of Ontology 3.0 implemented in CT, objects can be taken as types, with which we disagreed.

There seems to be a contradiction here: can we speak of typed objects or not? Well, one can speak of a type of an object either by looking at its trope composition or by examining $Hom(A, -)$ and $Hom(-, A)$ since types belong to morphisms and their compositions only. A type is not assigned to an object by intellectual categorization of humans, as done in the implementations of CT in Ontology 3.0. As a matter of fact, at the context level, it is not true that an object indicates its name. CT allows the examination of categories through arrows, functors, natural transformations, and universal constructions. On the one hand, thanks to CT's abstractions, many details are neglected, so deeper insights into various structures are gained. On the other, thanks to the ability of CT to objectify entities, especially concepts, as arrows in a category, combining concepts and propositions becomes a composition of arrows. That is to say, an

entity, such as 'movie,' can behave as an arrow in a category, as *is in* can behave like an object. Furthermore, there can be a morphism between *is in* : *physical* and *is in* : *conceptual*, or between ⌜BOOK⌝ of a context and ⌜BOOK⌝ of another context. We are not talking about a type system in that objects are classified in a sense we understood in Ontology 3.0. Again, types are about morphisms; since entities can be morphisms, we can talk about them being typed.

The dream/aim of Leibniz, Ada Lovelace, and other great minds have had that calculating a complicated argument with *philosophical algebra* has been searched into abstractions and objectification as expressions in morphisms. Nevertheless, this work offers a novel step in this endeavor and searches for the answer to how the machine knows the type of entities in advance at different levels.

4.6.5.2 Decomposition Levels in a Context

The most important feature of the urtrope theory is its ability to reduce everything to the same existential ground. This feature guarantees the semantic analysis of a context. For instance, think of a context category whose objects are propositions, arrows are typed morphisms, and one of whose objects is "a pen is blue." This proposition can be studied as an object of the hypothetical context, so it is studied by the arrows arriving at and departure from it. Figure 4.5 illustrates the contextual status of "a pen is blue," where where X, Y, Z, and W are propositions as well.

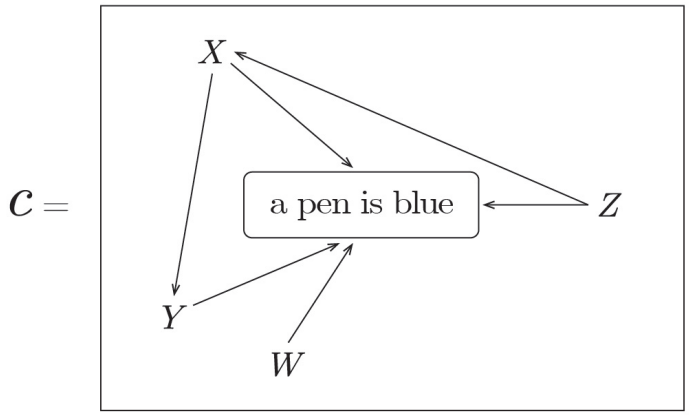


Figure 4.5: Contextual level representation of "a pen is blue"

When a more profound analysis is required, it is necessary to move to the propositional level representations. Figure 4.6 only shows the category whose objects are the source and target entities, and the arrow is the relation. Of course, the propositional analysis can be expanded so that other propositions of the context are also represented at the propositional level. So, functors and natural transformations can yield facts.

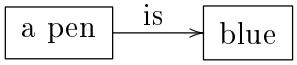


Figure 4.6: Propositional level representation of "a pen is blue"

The objects of a category in Figure 4.6 can be unfolded from the propositional level to the entity level. Since, at the propositional level, the arrow is an allowable/valid type so it can also be taken as an object at the entity level. That means the composition holds at the entity level, and each word of the proposition is taken as an object.[56] Beware that not only the objects but also the arrows of the proposition level are represented as categories as in Figure 4.7.

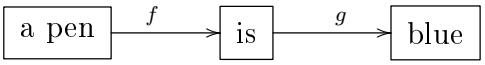


Figure 4.7: Entity level representation of "a pen is blue"

When every proposition of the hypothetical context is expanded to entity-level representation, then it may be possible to set up compositions between all entities in the context. Nevertheless, this is not the final level of representation. The penultimate representation is at the trope level. Each entity is represented as trope compositions; namely, each entity is represented as a topos. The trope level analysis is then between categories. Figure 4.8 illustrates hypothetical functors between categories.

Everything in Ontology 4.0 can be defined in terms of urtropes and beginning from urtropes everything can be constructed. To no surprise, the last level representation is at the urtrope level. At this level, all the constituents of the hypothetical context are decomposed, and type rules specific to urtropes are

[56] For the sake of simplicity, the indeterminate article is not separated from the noun.

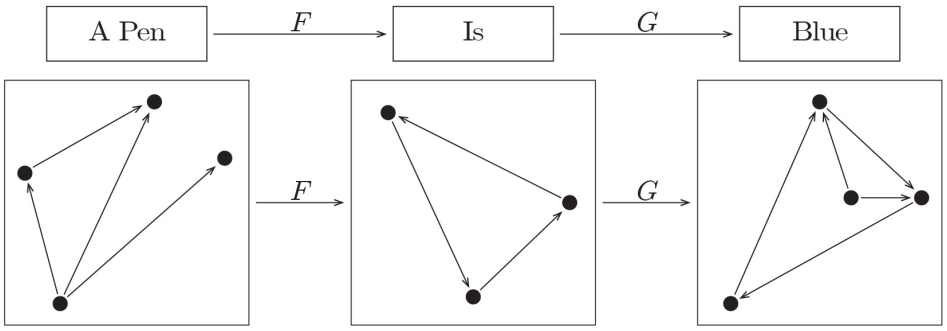


Figure 4.8: Trope level representation of "a pen is blue"

employed. Thanks to this level of analysis, the machine can be called understandable.

Just as the category `Type` of types is constructed, the category of `Proposition` of propositions and even the category of `Context` of contexts can be constructed. So, a theory can be an object of another theory; a proof can be a functor between two theories, or observation can be a category. Consequently, the entities of Ontology 4.0 can be studied on the base level, on higher levels, between these levels, or on the highest level, where knowledge is in its most abstracted form.[57] By doing so, a search space narrows down as the machine continues its investigation at detailed levels whenever morphisms attain their types without a doubt. Obviously, attaining types with a doubt is a statistical matter: the machine warrants the morphism type upon a trash hold, for example, between 90%-100%. Thus, whenever the morphisms get their types, `Type` is utilized to draw implicit morphisms. Note that a rigorous analysis of such categories is beyond this work. We omit careful examinations of the above since this work only aims to point out the need for a machine ontology and discuss and propose it.

These illustrations are not limited to some textual context. Consider the following example from Industry 4.0, for which relating the data coming from signals are of the most importance. Think of an application generated for smart cities that collects data from anywhere related. Let us narrow down the data collec-

[57] It would be a significant challenge to construct such a category that may be called the category of knowledge. Here is a try: `Knowledge` contains knowledgebases as objects and logical operators of any kind as arrows. The arrow compositions are possible worlds. Of course, that `Knowledge` is a category must be shown.

tion of the means of transportation where identified public transport cards are used without notification of the economic status of the card-owners. With classical tools, the data is clustered and then analyzed concerning some questions. However, the sensor data is ready to be queried in Ontology 4.0: the background knowledge is already in the machine. Who uses public transportation? Who uses identified public transport cards? What is an economic status? The machine knows the answers thanks to the urtrope representation of reality. Moreover, there would be no need to give a question to the machine. For instance, decomposing and recomposing tropes can relate entities and come up with facts only by raw video data, such that people who travel around 08:30–09:00 are in business attire; or more people travel before 08:30 than 09:30. Suppose the machine is asked "Why are there no children aged 15 using means of transportation at noon?" According to the visual data, the machine would say: that most of them are at school. That happens thanks to the machine's ability that independently relates so many entities, tropes, and urtropes at many levels.

4.6.5.3 Flexibility in Contexts

Flexibility in contexts means that the facts gained through the commuting diagrams of a context category are not rigid. Whenever something is changed in the context, new facts can occur. For instance, a new semantic property is attached to the trope composition of an entity, or a morphism type is changed. To make this concrete recall the previously given kitchen-tasks example.

In a kitchen,[58] when the task is washing the dishes, a coffee machine is out of the context, for washing has nothing to do with a coffee machine. In urtrope-theoretical parlance, *washing* cannot activate any semantic property of ⌜COFFEE MACHINE⌝. When the task is changed to making coffee, a dirty pan is not in the context for the same reasons mentioned above. In contrast, a dirty coffee mug may be an entity for making coffee-task, as a mug is needed for drinking coffee. Then, the process of washing becomes a constituent of this context. Again, in

[58] Think of 'kitchen' as a supercontext, where subcontexts interact.

urtrope-theoretical parlance, the semantic property of dirt of the 'mug' necessitates *washing* morphism to transform it into ⌜MUG⌝ so that the morphism can activate the appropriate semantic properties of the mug in the context of coffee making.

Clinging on this analogy, the machine can generate new facts when the context undergoes a change. Pay attention that any change in the context modifies the properties to be processed, not the entities. So, the machine must be able to determine the context by figuring out the semantic properties of entities and the relation types along with gathering information about objects by studying all arrows arriving at and departing from the object concurrently. That means, at the level of a context category, at least two dimensions work simultaneously: a context category where objects are processed through typed arrows and an entity category where all the objects and the arrows of the context are penetrated with a consideration of different categorial constructions.

However, we do not want to sound as if we *already* know the mathematics behind these constructions, but we *do* know that all of them are feasible and tenable. Typing rules and urtrope-theoretical axioms tell us the arrow composition of a fact, an entity, a semantic property, or even a trope. There is always a way down to the base level, urtrope level, and a way up to higher levels, which are constructed through objectifying categories as objects. That is to say, CT allows decomposing and recomposing structures; as Marquis (2021) puts it, "once a type of structure has been defined, it is imperative to determine how new structures can be constructed out of the given one [...] [and] how given structures can be decomposed into more elementary substructures." Thanks to the tools that topos theory provides, the machine can go as high as possible and as deep as to urtropes; so that, it can organize and layer structures with ease. Hence, category theoretical innovation in terms of the structure lies in the composition and decomposition, if possible, of the structure whose type has been defined. That how structures of a certain type compose new structures and can be decomposed lies at the heart of CT (Marquis, 2021).

4.6.5.4 Figurative Meanings in Contexts

Think of that one is said to think of 'yellow,' 'lemon,' 'crying,' 'yellow lemon,' 'crying lemon,' or 'crying yellow lemon.' Without going into the diversity of opinions about or the depths of the views on human cognitive processes, we can say that all these entities trigger the human mind separately, causing some mental formations. Humans can imagine a crying yellow lemon picture or even start salivating when hearing the word 'lemon.' Furthermore, a human can understand that 'crying lemon,' 'crying yellow,' and 'crying yellow lemon' are used figuratively. They do not think that crying is a semantic property of a lemon. On the other hand, in order for the machine to understand, the entities must be given in a context; viz., neither 'lemon' nor 'yellow' corresponds to anything in the machine. In other words, 'yellow lemon' is understandable since the composition of these entities creates a context. The same holds for 'crying lemon,' 'crying yellow,' and 'crying yellow lemon.' However, how can the machine figure out that the last three are figures of speech?

A syntactical analysis gives that both 'yellow' and 'crying' are adjectives. However, the type of morphism between 'yellow' and 'lemon' differs from the one between 'crying' and 'lemon.' That is, yellow as an adjective has a structural correspondence to 'lemon,' just like 'fresh,' or 'chopped;' whereas a property, viz., a trope, of 'crying' is attributed to 'lemon,' that property does not correspond to the trope compositional structure of lemon. Thus, a structural correspondence is a morphism type between two entities, which can be constructed as a semantic property of either entity. In our case, 'yellow' can be structured as a trope composition in 'lemon.' On the other hand, figurative correspondence is a morphism type between two entities, which cannot be constructed as a semantic property of either entity. As such, propositional- or entity-level decompositions provide to find structural-semantic properties of entities so that the morphisms of figurative types can be detected.

Another example of structural and non-structural relations is 'blue pen' and 'blue voyage:' in the former, 'blue' has structural correspondence to 'pen,' the correspondence is known through the tropic composition of 'pen;' whereas in the

latter, 'blue' stands with some figurative meaning, which does not correspond to the structure of 'voyage.' The question arises: does any adjective connected to the noun 'voyage' form a figure of speech? The answer is negative, although it is syntactically affirmative. For instance, there cannot be a morphism between ⌜VOYAGE⌝ and ⌜PET⌝ or ⌜ODD: integer⌝. So, a figurative morphism type refers to one of the structural correspondences of an entity. For instance, 'blue' is a structural correspondence of 'sea;' ⌜SEA⌝ has 'blue' as a semantic property. So, the meaning of 'blue' is transferred through 'sea,' then, a blue voyage refers to a sailing vacation, or more precisely, it is *the* vacation along the Turkish Riviera. Alternatively, a purple voyage can refer to a stroll through the lavender gardens, as purple is a semantic property of lavender.

To sum this part up in one sentence, categorial/topos theoretical representations of contexts provide frameworks in which figurative meanings can be drawn. All these examples show us that CT enables defining relations between entities and relations, which is not possible in the previously utilized frameworks of Ontology 3.0. Thus, in Ontology 4.0, morphisms between entities, between relations, between contexts and relations, between relations and entities, between facts and relations, and alike can be defined. Ontology 4.0 is the most potent representation and form of association so far we examined.

4.7 Conclusion

The limited representation of ontology systems in Ontology 3.0 operating only a fraction of the world took us on the journey to search for a machine ontology. We have also witnessed that upper-level ontology studies, which are at the forefront of efforts to ensure the presentation of background information, cannot present the diversity of representations in the field of logic and again offer a limited representation as they are based on human power. Based on all of these, we claimed that it is necessary to move the being into a formal field –phenomena into data– and that this would be possible with a relationship-based approach. Thus, standardization would be provided as well as formally representing the being in the machine. We have demonstrated the ontological basis of this structure,

namely of Ontology 4.0, with the urtrope theory. What remained, then, was to show its computational basis. We have determined the computational basis of Ontology 4.0 as the category theory since it is the formal structure representing the urtrope theory. In light of this, in this section, we examined both category theory and how it would formalize urtrope theory. Knowing what category theory is and how it is used in ontology has an important place in our research. Therefore, at the conclusion of this chapter, we would like to include why CT is necessary both today and future of informatics and in our study.

CT is mainly regarded as the new foundation for mathematics, although it is controversial. As such, a category is an abstraction of a mathematical theory so that it can model a mathematical theory, such as the theory of sets or the theory of rings (Biss, 2003). Moreover, a category encapsulates fundamental aspects of some mathematical fields, such as Abelian categories encapsulate the structure of homological algebra (Marquis, 2021). Being a *lingua franca* between mathematics fields, CT also relates, compares, and contrasts categories, so the distinct mathematical fields. This most magnificent power of CT springs from its ability to model a mathematical theory in terms of its structure. Take the first example of CT: the relationship between algebra and topology is shown thanks to specific functors that relate the structures among real numbers and topological spaces, and continuous functions and continuous mappings. Or, the category of sets and partial functions is shown to be equivalent to the category of pointed sets; the category of finite Boolean algebras is shown to be equivalent to the opposite of the category of finite sets (Awodey, 2006).

The competence of CT is not limited to mathematics, and it also stands out in science. Spivak (2015) provides some examples of applications of CT throughout science. For instance, CT is used for modeling the relationships between geometry and algebra in mathematics, functional programs in programming language theory, the hierarchy found in protein structures in materials science, the relationship between local and global behavior in various field theories in physics, the symmetry groups of crystal structures in crystallography. Moreover, Spivak (2015) reports that the National Institute of Standards and Technology (NIST) recognizes CT as a potential mathematical foundation for Big Data; for instance,

CT "as applied to sensor-actuator data in the Internet of Things, or as applied to supply chains which must account for multiple levels of granularity." On the one hand, each database has its own aspects and ontological commitments that bring out some theories; on the other hand, each programming language has its own aspects and ontological commitments. Queries, data migration, optimization, and others suffer from the discordance between database schemas and programming languages (Spivak, 2010). For instance, how the NULL values are defined is up to programming languages, and the implementations of those values in databases create a mess. It is plodding, if not impossible, to merge different databases written in different languages. Understanding a schema and migrating data in it is difficult, prone to errors, and quite tedious. Nevertheless, CT brings out a unity between programming languages and representing data.[59] CT fulfills the need for a single language and protocol for a schema. Therefore, CT serves as a firm foundation for semantics in linguistics and programming languages in a "powerful, expressive, and scalable, yet axiomatically simple" way (Spivak, 2010). Note also that since data and their representation are unified, and morphisms preserve structures, there is no information loss. Ontology 4.0 solves the problem of structuring data but also brings about a unity between programming languages and data management. That is to say, CT is not just a formal system that best suits the urtrope theory. Just as the urtrope theory is the best ontological approach for a machine ontology, CT is likewise the best representation system of all computational processes concerning the machine. So, CT is handy not only for representing data but also for programming languages, protocols, and other data processing techniques.

The marriage of the urtrope theory and category theory above all gives us the possibility of phenomena into data realization. Accordingly, everything becomes representable by relations through contentless building blocks. In other words, all beings become data, and all the data are further represented by the same ontological type, viz., urtropes. Since everything is represented on the same

[59] An application of CT to programming languages is called *functional programming*. Functional programming languages as categories are the key aspect of representing and implementing everything with a single tool. See Harper (2016) for details.

level, there is no longer the separation of source, target, and relation as in philontologies: a target can be processed as a relation, and a relation can be processed as a source. In fact, even the processing of operators is now at issue since the logic operators can be reduced to category theoretical ground. In addition to these, the semantic properties of entities become critical according to the context where they are.

Thanks to the decomposition and recomposition ability of urtropes, static role assignment is not a destiny anymore. It is dynamic in the sense that there is an ability to create an exponential representation space. Thus, this marriage, called Ontology 4.0, seems to be the key to realizing Science 4.0 and Web 4.0 since CT is a tool to provide new knowledge when inference is defined as manipulation of the structured data to produce new formula (Biss, 2003).[60]

At the end of this chapter, we can easily say the following: It is possible to construct Ontology 4.0, and structure-analysis-interpret-and-inference sequencing, which we claim to be necessary for machine-understanding, can be achieved thanks to formalizations of CT. Ontology 4.0 is the very ground on which contexts of ISW 4.0s can be processed. Bringing everything down to the same ground in terms of contentless constructors, which is its trademark, indicates a system where ontological classification and computation are one and the same. Thus, Ontology 4.0 both represents and processes the phenomena in the machine.

[60] However, the impact of Ontology 4.0 on Industry 4.0 may be in the shadows of knowledge representation. To our understanding, knowledge representation is not limited to contextual data; rather, it includes Data 2.0, but its understandable version.

CHAPTER 5

CONCLUSION: FEATURES OF ONTOLOGY 4.0

The conclusion of this dissertation is threefold. The first part will summarize what has been done and will be an introduction to the second part, where the features of the product of this dissertation, viz., Ontology 4.0, will be explored. Lastly, we will mention some future works following this work.

5.1 The Road to Ontology 4.0

One of today's most striking and top-raking topics is undoubtedly Big Data and its contributions. At the beginning of this dissertation, we conferred the promises of Big Data over three significant areas at the center of our lives. We discussed how the industry would undergo a new paradigm shift with Big Data. This paradigm shift is smart factories, where all the machines are interoperable; they can communicate by speaking the same language. We claimed that Big Data and the machine would create a similar paradigm shift in science, where the machine would act as if a colleague would be able to analyze all previously produced scientific data, make observations, and develop hypotheses using the scientific data previously produced with these observations. Next, we discussed the necessity of finding a solution to getting lost among the millions of webpages. We mentioned that if there were a paradigm shift in how the Web worked, it would start with classifying all existing contents according to the searched terms. Finally, we claimed that the machine must be autonomous in order to obtain these paradigm shifts that would guarantee the 4.0 versions of each domain. We declared that the common feature of the way to realize ISW 4.0s was that

the machine could comprehend existing entities in contexts with their possible interactions and make inferences from them.

5.1.1 What Needed for the Realization of ISW 4.0s

Once again, the common feature of the way to realize ISW 4.0s was that the machine could comprehend existing entities in contexts with their possible interactions and make inferences from them. That is what we mean by *machine-understandability*. We say a human can *understand* a language when they can put some words together by obeying the grammar rules. In other words, understanding means knowing what to do with the pieces of something in a new context; the pieces are numbers when the new context is calculation; the pieces are words when the context is a conversation; the pieces are keys when the context is playing the piano. In this respect, the machine needs to be inferential: it needs to structure and decompose the given situation, interpret the structure and decomposition, and then make inferences. When something can do all these processes, it is called *autonomous*. That is to say, the machine needs to be autonomous in order for there to be a paradigm shift in Industry, Science, and Web.

An autonomous agent can respond the new situations by considering all the possible meanings and usages of the pieces of the given situations. But, what about the machine? How can it consider the entities in a context? Recall that it is through the *semantic properties* of entities. An entity is in a context with some highlighted semantic properties. For instance, the semantic property of intelligence is essential in the context of "Laura is a Nobel laureate;" this property has nothing to do with the context of "Laura eats bread." When the machine *knows* the semantic properties of entities and, *selects* some of them particular to a context, and *processes* them, then it is called to behave autonomously.

The questions to be answered are (1) how the machine can know and (2) select the semantic properties and (3) how it can process them. The answers: (1) the machine knows the semantic properties of entities by the representation of them: all the entities are represented in terms of their semantic properties.

(2) The machine highlights the right semantic properties of entities by their interactions in a context. (3) It can process them in a relation/process-based formal framework. Let us remember how we have reached these answers.

5.1.1.1 Representation in terms of semantic properties

As a matter of fact, what we are trying to do is structure entities. Because dealing with Big Data is dealing with unstructured data in an open system. ISW 4.0s also get their fair share of this. At this point, two solutions can be proposed with existing technologies: (1) structuring data or (2) processing data without structuring them. Our research has shown that these two approaches are unsuitable for processing unstructured data to realize ISW 4.0s. Let us explain this situation by adding new ones to our research and begin with ontologies, a tool for structuring data.

The homepage of the website of Cyc opens with a catchphrase "Cyc: Logical Reasoning with the World's Largest Knowledge Base" (Cycorp, 2018). Initiated in 1984, Cyc has become one of the leading upper-level ontologies in information systems. Behind this success lies laborious determination to represent *everything* and master machine intelligence. On the way to such a huge success, Douglas Bruce Lenat, the father of Cyc and the CEO of Cycorpt, and Ramanathan V. Guha, one of the co-leaders of the Cyc Project, wrote a book called *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project* in 1989. Of many prominent contemporary computer scientists who reviewed the book, Drew McDermott (1993, p. 58) wrote the following:

> If we want to be able to represent *anything*, then we get further and further from the practicalities of frame organization, and deeper and deeper into the quagmire of logic and philosophy. It seems impossible to get an overarching scheme of representation without solving problems like the nature of causality [...], identifying events in different possible worlds [...], the relations among intentionality, rights, and duties [...], the distinction between voluntary and coerced actions [...], the distinction between de re and de dicto belief [...], and the theory of defeasible reasoning [...]. What's frustrating is that the project of representing consensus beliefs seems to require the assumption that everyone is already in possession, at an unconscious level, of solutions to these knotty.

Although Cyc is released as if it can structure any data set, even today, it could not have escaped from the old critiques. Along Cyc, many other on-

tologies are claimed to solve the abovementioned problems. The main problem with the label-based approach is that all these constructions are hinged upon set-theoretical approaches, and all of these ontologies, either philosophically grounded or modeled for commercial/computational purposes, reflect human understanding, and such reflection is converted into machine 'understanding.' However, we claim that there must be a shift towards constructing machine ontologies, which serve for the *machine intelligence*. Machines and humans are of different categories, so as their intelligence and their intellectual agency (Zambak, 2014): the way humans interpret the world is different than the machines do or can do. However, studies in machine intelligence, or artificial general intelligence, are focused on how the machine can perform just like humans capture the phenomena, understand the world, communicate with their environment, employ reasoning and solve problems, and perform other intellectual operations. Along with these, such *overarching schemes of representation* is impossible to construct since there is always something, a scene, a belief, a relation, an event, left to be represented. That is to say, it is impossible to represent every actual and possible context; consequently, the machine cannot know the semantic properties of an entity that has not been labeled until now.

So, if structuring data through ontologies is impossible, what about the machine's ability to process data without specifying or structuring semantic properties? Here, we would like to survey this investigation through statistical methods, especially machine learning techniques. There are four impediments that statistical models cannot remedy ISW 4.0s. The first one is that current machine learning algorithms lack the ability to identify and respond to new circumstances they have not been trained for. That means the alleged "autonomy" of the machine cannot be achieved through statistical methods. The second impediment is that a constructed system cannot explain its choices. We cannot get an answer to "why did this autonomous car accelerate instead of break?" The black-box nature of these systems is not good fellows in industry and science. The third one is that these models require *big* data to be constructed, and the amount of the necessary data is hardly detected. For instance, GPT-3's training data size is about 45TB, and 175 billion parameters are used (Radford, Wu, Child,

et al., 2019). If there were more data and parameters, could GPT-4 perform better? Even if it could, it cannot escape from the second impediment. The last obstacle is the lack of understanding of cause-effect connections. Nevertheless, Pearl (2019) purport that his *structural causal models (SCM) framework* could overcome impediments of statistical models; for instance, it allows transparency and testability, and causal discovery. However, the SCM framework works with structured data, yet "the theoretical limitations of model-free machine learning do not apply to tasks of prediction, diagnosis, and recognition" (p. 60). To sum up, statistical methods are crucial in data science, and no one can deny the powerful results that machine learning algorithms create. That said, structuring data is the real impediment before machine-understandably and making inferences in open systems. Once the unstructured turn into structured, these methods can produce much more successful results, even in open systems.

Then, we came to the idea that if processing semantic properties are at the core of machine understanding, entities would be represented from the beginning as a collection of their semantic properties. Before moving on to the characteristics of such a representation system, let us answer the second and third questions.

5.1.1.2 Gaining meaning through interactions

As an autonomous agent, the machine can highlight the right semantic properties of entities by their interactions in a context. That means the machine knows which semantic properties are active when at least two entities interact. In old fashion triple method, RDF, for instance, semantic properties are attached to entities as metadata which determines the ontological statuses of the entities. The inference rules of OWL, for instance, operate on the semantic properties, and so on the entities. If the entities are to be represented as a composition of their semantic properties, the machine will go through the selection of the semantic properties by semantic types. Semantic types will assign the legitimate interactions gained from the relations in a context. In other words, an entity is not assigned a type, but its semantic properties have types. Then, we can say that if the entity has a semantic type, that is, a semantic property, it is in a

relationship because we cannot talk about a semantic property of an entity by itself. Whenever an entity in interaction a semantic type occurs, when a semantic type occurs, the ontological status of the entity will emerge. Therefore, there is no need for inference rules like RDFS or OWL.

Zambak (2013) notes that such an agent has the ability to "choose and shift different set of rules according to its situation." The machine can choose and shift the semantic properties of entities according to contexts. That is to say, the changing situations occur in contexts where different semantic properties, accordingly, different semantic types are chosen to be processed. Thus, the machine will determine the semantic types through the relations in a context, and this determination is the process of highlighting the right semantic properties.

5.1.1.3 Process-based computation

On the way through autonomous machines, entities are represented in terms of their semantic properties, and the inference system is embedded into transitions through semantic types. Such representation requires a different approach to computation. Most of the representations that has been implemented so far has prioritized the entities. Both representation and computation are modeled in an object-oriented fashion, but now there is a need for a framework of representation and computation where the relationship takes priority. Hence, semantic properties must be processed in a relation/process-based formal framework.

5.1.2 Why a Machine Ontology Needed

All the ontologies in Ontology 3.0 are context-dependent; hence, the machine cannot cross the understandability barrier. It is doomed to be called a "reader." To create a machine that understands, there needs an ontology that automatically organizes the entities in a context and prepares semantic properties to be ready for processing. Besides, this machine ontology should be a framework for automatically constructing domain ontologies. It is not for satisfying human intellectual curiosity but rather for providing an ontological infrastructure nec-

essary for machines to be autonomous. Building a machine ontology approach is also a paradigm shift in knowledge representation studies.

As supported by philosophy and quantum mechanics, whose evidence was provided in this dissertation, relation-based representation portrays reality. In light of and courage from these, we adopted a trope theoretical approach for the philosophical basis of the machine ontology we wanted to construct. Tropes, in the philontological sense, are properties whose compresences establish entities. The adopted version of trope theory in our machine ontology says tropes are semantic properties whose compositions give rise to entities. By doing this, we have laid the representational foundations of machine ontology.

5.1.3 The Role of Typification in Processing Tropes

Since the representation foundations of machine ontology have been laid, it has now come to how this representation system will be processed. We just talked about how semantic types would work in the previous paragraphs. Therefore, a type theory does the job. Types are abstractions of representations that can help the machine to specify the operations for a collection of representations. Semantic properties become processable by assigning types to tropes and formulating type rules. For this reason, we examined in this dissertation that a system that operates the tropes on the machines is type theoretical.

5.1.4 A Must: Contentless Building Blocks

We can also describe the machine's being autonomous as follows: the ability to process contexts. For example, let us consider a doctor and a nurse in an examination room of a hospital. The job descriptions of these people are clear. However, let us think of such a context that the place and people are the same, but this time the nurse is the patient, and the doctor is taking care of him. The way these two people relate varies as the roles change in contexts. So, when the semantic types change, the meaning of the context will also change.

It seems that the representation and process we have said so far can be more or less in Ontology 3.0. However, the processing of semantic properties should not end there. No ontology of Ontology 3.0 can process semantic properties of semantic properties. In other words, types also need to be processed to make the representation system closer to reality. Suppose we have two propositions "All humans are mortal" and "Socrates is a philosopher." A human can conclude "All philosophers are mortal" or, more precisely, "Socrates is mortal." Once that 'philosopher' is of a human type is known, these two propositions can be merged through a meta-type of 'philosopher,' viz., 'human.' So, the machine is not only capable of processing different semantic properties of entities in an identical setting but also processing types of types. Thus, not only semantic types but also their types must be computable. Said that type theories are notoriously incapable of representing this structure.

Although a type theoretical approach is essential for a machine ontology, adhering to this approach will cause problems in automatically creating types of types. We must modify our type theoretical approach because there are two crucial impediments to applying type theories. The first impediment is that type theories commit infinite regress. A trope can be represented with another trope, then that trope with another one. We always have to come up with a new trope to represent another, which cannot be calculated. To break this, we can say that it comes to a point where it becomes self-reflexive. However, there is no self-reference in type theories from the beginning. So, it is not out of type theories that types of types can be constructed without falling into infinite regress.

In fact, when we look at computer science, we witness that contentful things are produced from contentless ones. At its simplest terms,0s and 1s represent numbers, letters, and even operations. Untyped formal systems, which have an important place in computer sciences, will guide us. So, to summarize, we cannot use type theories as they are because types are always contentful, and their extensional nature causes infinite regress in calculating the types of types. As semantic properties are specified from the beginning in the representations, other semantic properties of entities are sacrificed. This creates losses in rep-

resentations. The targets of ISW 4.0s cannot afford such losses; as such, the types of types must be constructed by a machine ontology. Therefore, we need to represent entities with semantic properties but their types with contentless units.

Will reducing the entire machine ontology to contentless units allow us to benefit from the blessings of type theory and get rid of its troubles? We know from the untyped theories that their combination will give the types. So there is no obstacle to creating types of types, which is actually the creation of semantic types. Because a type's type is also a type, and this is a semantic property. In the dissertation, we showed that what we call untyped units is the ontological being that creates everything. Just as semantic properties come together to form entities, their coming together will create typed beings. In that case, their infinite composition will lead to the formation of an infinite number of types. Thus, contentless building blocks allow the type system to be modified without looping and self-reflexive and saves the establishment of a machine ontology.

5.1.5 Ontological Foundations: The Urtrope Theory

We called the contentless building blocks *urtropes*, and the ontological foundations of the machine ontology the *urtrope theory*. Borrowed from German, the prefix "ur-" means that earliest form of, primitive, or original. In the urtrope theory, urtropes are contentless building blocks of representation, tropes are compositions of urtropes, and entities are compositions of tropes. Thus, the machine can represent any data in terms of urtropes.

The urtrope theory allows self-reflexivity; namely, an urtrope can be represented by other urtropes. But how can an urtrope or a composition of some urtropes represent an urtrope? The solution we offer is typifying identities. This system works as follows: urtropes are classified according to identity types, and at another level, an identity type is typified by other identity types; viz., subtypes of identity are created. The urtropes in the question are represented at the different levels of identity types. That is to say, an urtrope, say, u_a is represented by u_b, since they are identical in type t_1. The urtrope u_b is represented by u_c

since they are identical in type t_2. Lastly, the urtrope u_c is represented by u_a since they are identical in type t_3. When this loop ends and if the typing rules allow the transactions through these types, we can say that u_a is represented by u_c. Moreover, in light of this, we can generalize that a machine ontology based on the urtrope theory is also self-reflexive. Again, an isomorphism between two distinct trope compositions of identity types guarantees that the system is self-reflexive.

5.1.6 Computational Foundations: Category Theory

After introducing the urtrope theory, we need a formal theory for implementing the machine ontology. As we have noted before, although there may be some other formal systems to fit the urtrope theory, we chose category theory to formalize the urtrope theory. The motivation behind our choice is that both theories share the same foundations: the objects are contentless, and all the computations take place over the relations.

Our research led us to formalize urtropes with C-monoids-without terminal object, and the rest of the machine ontological categories –tropes, entities, contexts—with toposes. As category theory (CT) was not introduced with any urtrope theoretical intensions, we also noted that some axioms and rules must be introduced into the employment of CT.

Although controversial, CT is mainly regarded as the new foundation for mathematics. Being a *lingua franca* between mathematics fields, CT relates, compares, and contrasts categories, which refer to distinct mathematical theories. So powerful in mathematics and so needed for the formalization of the urtrope theory, CT has gained importance in science and data science. Spivak (2015) reports that the National Institute of Standards and Technology (NIST) recognizes CT as a potential mathematical foundation in the era of Big Data; for instance, CT is applied to sensor-actuator data in the Internet of Things, supply chainsthat account for multiple levels of granularity, and functional programs in programming language theory. We hope that CT will catch the attention of information systems.

5.1.7 Ontology 4.0 as a Machine Ontology

So far, we have just mentioned the most important headlines of this work, which covers all the substantial research and contemplation collected under the headlines. So, we will mention what an ontology Ontology 4.0 is.

Ontology 4.0 is such an ontology that, like all ontologies, it provides standardization in representation. It does this standardization with the power of representing everything as urtrope compositions. Ontology 4.0 also provides automatic structuralization. In other words, it gives the ontological structure, such as which semantic properties are activated, what are the types of interactions, and alike. In fact, this means that Ontology 4.0 is an ontology that generates ontologies. Although it can be compared to upper-level ontologies at this point, Ontology 4.0 has the power to represent itself. No ontology of Ontology 3.0 can represent itself in its own representation language. Hence, Ontology 4.0 is the only ontology that presents both itself and the ontologies of all contexts. Additionally, Ontology 4.0's power to handle syntax and semantics on the same plane differs from all other human ontologies. That is, representation and inference are in the same framework.

Ontology 4.0 is such an ontology that it is not limited to natural language studies. When knowledge representation is mentioned, it can be thought that it includes only Science and Web, but Industry is also involved. Although statements produced in Industry have different epistemic types than statements of Science and Web, knowledge representation includes machine communication and interoperability. Thus, Ontology 4.0 can represent not only textual data but also models, data types, data structures, and others.

This part of this chapter just skimmed the course of this dissertation, where each part was explained in detail in a corresponding chapter. The second part of this chapter will examine the features of the machine ontology, aka. *Ontology 4.0*. These features exhibit the prowess of our theoretical construction. We believe that the following part, where the basic and beyond conditions of/for machine-understandability are listed, is the actual conclusion of this research.

5.2 The Features of Ontology 4.0

This part of the conclusion explores the features of Ontology 4.0, which are crucial for crosschecking our thesis. While discussing those features, we will see how Ontology 4.0 meets the needs of the realization of ISW 4.0s.

Philontologies, Ontology 1.0 and 2.0, seek to provide a complete description and explanation, and exhaustive classification of all entities in all spheres of beings, Ontology 3.0 seeks to provide a standardization of entities and relations so that knowledgebases are created for standardization, information retrieval and extraction, and knowledge extraction. However, we unfolded that Ontology 3.0 is incapable of the demands for the realization of ISW 40.s, whose features were explored in the chapter Autonomy and Data. Moreover, studied them, we purported that the approach for representation should be shifted towards a new paradigm in which *being* and *data* must be orchestrated for the machine; namely, an approach that sues data should be inherently represented in a machine ontology. As far as Ontology 4.0 is claimed to be a machine ontology, in the following paragraphs, let us first recall the most crucial features of ISW 4.0s and then explore the features of Ontology 4.0; the features that deal with the features of its environment belonging to ISW 4.0s.

The very first feature of ISW 4.0s is their being open systems. A system is open if it is interactive and vice versa (Wegner, 1998); that is, what determines a system's being open is its external interactions with its environment. Smart factories are open systems, as the machine manipulate data coming from various sources in real time; scientific exploration is a process of an open system, as it requires examining a phenomenon from different settings that reflect myriad interactions; the Web is itself an open world; thus, any work that encapsulates the Web as a whole must operate among various sources with different structures. Note that ISW 1.0s-3.0s have been dealing with the issues of a domain whose structure is defined in advance of the machine operations; on the other hand, ISW 4.0s need to deal with the issues of a domain whose elements are in interaction with other elements of other domains. That is to say, indeed, knowledge of domains is incomplete *per se* and roles/behaviors of the entities in a domain

can vary. This incomplete and contingent nature of ISW 4.0s is related to their feature of being open systems.

The second feature of ISW 4.0s is being dynamic and evolving. As they are interactive, each interaction creates a dynamic environment; with each interaction, these systems change. For instance, a new machine sends particular signals to other machines, which are to be interpreted in a novel way; or new concepts can be introduced in a field, with new connections must be figured out. Note that interactions are not linear necessarily; they can be concurrent. Thus, the dynamicity and evolution of ISW 4.0s are directly related to the representation of concurrency.

The last feature we would like to mention is that ISW 4.0s are complex systems in terms of interaction and, of course, in terms of computation. The more there are interactions, the more there is interactive complexity (Wegner, 1998). The interactions of concern are from the data level to the context level: the machine cannot limit its data manipulation capacity to a restricted domain but must upgrade it for the intercontextual level. Thus, the machine has to manipulate data with distinct roles/behaviors that arise from dynamic, nonlinear, and emergent interactions. For instance, traffic is a complex system by its nature. A task in Industry 4.0 integrated with traffic has a complex component that makes the task complex. Hence, the machine should not only deal with the complexity of components, but also the complexity that emerges from the interactions between components.

In a nutshell, the features of the environment in which Ontology 4.0 survives can be summed up to be open, incomplete, contingent, dynamic, and complex. Now, let us explore the features of Ontology 4.0 and analyze which deals with the issues of this environment.

5.2.1 A standard of representation

In the absence of the machine ontology, data standardization has been based on other data, called *metadata*. Even standardization of standardization might be

required since each standardization gives one extensional aspect of the entities, which have various aspects that gain significance in other domains. On the other hand, Ontology 4.0 provides standardization of representations thanks to non-extensional urtropes that are *the standardization* of data: entities and their interrelations are represented independently from any single world state in a formal fashion that deals with them on the same level. These untyped building blocks of Ontology 4.0 guarantee the standardization of representation in the machine.

It is worth mentioning a caveat. That urtropes can be represented by other urtropes in circularity may be thought as if urtropes were represented on standardization of some urtropes. However, that urtropes are representable by other urtropes does not lack their non-extensional aspect; it does not give them an extensional status. This very feature of urtropes that is representable by other urtropes provides many levels of urtrope compositions. Thus, the urtrope level is the standardization level, at which everything becomes comparable.

5.2.2 A unification of syntax and semantics

Traditional semantics depend on the interpretations of the logical constants important in reasoning. As such, meanings of propositions and logical constants are inhabited in their interpretations, such as in Tarskian semantics, where $\wedge$ means *and* and $True \wedge False$ is false, for instance. Accordingly, semantics hinge on external rules specified in syntax. This approach, however, comes along with hindrances, e.g., one cannot track what can be computed and cannot be. Accordingly, and furthermore, one cannot execute definitions following the traditional semantics perspective (Constable, 1991). For instance, Dedekind cuts and Cauchy sequences both define real numbers, but neither is computable. This situation shows that even such basic concepts, which frequently arise in computer science, cannot be related to each other. That is to say, traditional syntax-semantics distinction may cause computational impossibilities. Thus, there must be a system in which syntax and semantics are unified, and the constructions are computable. Indeed, the introduction of Martin Löf's type theory as a

foundation for computation guaranteed the existence of such a system. Martin Löf's type theory makes definitions computable. Thus, semantics is located within the inference system, and computability is prior to definitions. That is to say, defining external rules for interpretation is ruled out from semantics computation (Cf. Constable, 1991), and both logical and extra-logical definitions are intrinsic to the inference system so that computation is just a part of the theory (Schroeder-Heister, 2018).[1]

Lambek and Scott (1988) show that a category itself can be taken as an abstract proof theory, such that an arrow $A \longrightarrow B$ in a category $\mathcal{C}$ is an abstract from of B from A: this representation is more than the derivation of B from A since the arrow has its own epistemological and ontological status (Schroeder-Heister, 2018). Moreover, as shown by Došen (2003), the identity of arrows means the identity of proofs; thus, proofs are not just vehicles to establish consequences, but at the same time, they have their own existence. Let us unpack this: The Curry-Howard correspondence suggests propositions-as-types, according to which a proposition P has a certain proof that can be constructed as a certain term t is of type P. That is, the proposition P is identified with type P. In order to show that t is of type P, that t is a proof of P must be shown. In Martin-Löf's type theory, there are twofold senses of proof (Schroeder-Heister, 2018). Firstly, there are proofs of statements of the form $t : P$, in which the term t represents a proof of the proposition P. Secondly, proving $t : P$ is showing that t is a proof (object) for P. The argumentations are done in the first sense of proof, whereas in the second sense of proof, the meanings are explored.

Topos theory provides perfect execution in the machine, as its structure equates symbolic expressions and their inferences (Cf. Galmiche, 1990). So, the twofold sense of proof has its machine ontological status and innate epistemic value. Consequently, thanks to that theory of urtropes that are the building blocks of a typed categorial system where topos theoretical constructions name the types, Ontology 4.0 is a unification of syntax and semantics.

[1] What has been said is all about *proof-theoretic semantics*. Thus, Ontology 4.0 is a sort of proof-theoretic semantics.

As a final note, we would like to turn back a game we have proposed that is as follows. One who does not know a single Turkish word is given a Turkish text. They are supposed to operate the text with the help of the machine. Ontology 4.0 is the best tool for this task: the person writes the text, which is nothing but a bunch of symbols to the machine. However, those symbols carry some meaning along with themselves: operations on the symbols are just inferences. That shows that Ontology 4.0 is for what ISW 4.0s long.

5.2.3 A process-based framework

Understanding a mathematical structure requires understanding the process of preserving this structure,[2] and CT formalizes a relation between structures and the processes preserving these structures (Singh, Isah, & Ibrahim, 2012). That is to say, a representation in CT is a formalization of processes. So, Ontology 4.0 offers a process-based framework. Let us see how it does it.

Consider the calculation $\frac{1+2\times3}{4-5/6}$. The result indeed depends on the order of relations, namely the process. Mathematicians agree on the order of operations, abbreviated as PEMDAS: Parentheses, Exponents, Multiplication, and Division, then Addition and Subtraction–from left to right–, in order to expel computational disambiguation. So, the order of processes is of the greatest matter. For instance, one may know the ingredients in a dish, the equipment used, and the relationship between the ingredients and the equipment. In Ontology 3.0 parlance, one knows the entities and relations. They can only cook the dish when they know the order of the relations. Only the order of the processes preserves the structure.

Ontology 4.0 offers a framework in which everything is represented in terms of relations. The type forming rules determine the order of the trope compositions. That is, the typing rules construe more than a relation-based framework: a process-based framework. Although it is totally legitimate to name this frame-

[2] This idea belongs to Amalie Emmy Noether, an influential figure in mathematics, especially in abstract algebra and CT.

work as a process ontology –relations represent everything, the computations are only on relations–, a caveat is needed: process ontologies or process-like ontologies in Ontology 3.0 prioritize relations over entities, whereas Ontology 4.0 has no such dichotomy at all: urtropes construct the world. A final note is that Ontology 4.0 is *not* a process-based system itself; instead, it builds process-based systems.

5.2.4 A dynamic and evolving system

Ontology 4.0 is dynamic and evolving, as it stands by alteration, transformation, and update. The feature of dynamicity arises from the fact that urtropes are reorganizable: their different compositions give rise to different semantic properties in different contexts and types. At this rate, the reorganisability of urtropes and even of tropes–as their different compositions give rise to different entities, and so different types–, enables them manifest potential compositions. It is like wood is potentially a table and potentially a picture frame; what makes it a table or a picture frame depends on the workshop. So, depending on the context, an urtrope/a trope actualizes its potential compositions. However, the contextual roles/behaviors of urtropes/tropes can be changed when interacting with other urtropes/tropes. That is to say, unlike wood, once they actualize their roles/behaviors in a context, there can still be unactualized, viz., potential, interactions that can appear when something new-a trope, a morphism, an object- is introduced to the context; because what determines it to be dynamic is the knowledge of which potential interactions can be established under what conditions. Hence, urtropes and tropes are dynamic, so as Ontology 4.0.

That said, recall the "strategy of relativization" that paves the way for dynamic representations. Krömer (2007) utters this strategy by highlighting that being an object, an arrow, a category, or a functor depends on the relations between things: a category can be an object of another category, or a natural transformation can be an arrow of a category. Thus, the changeable statuses of trope compositions also represent the dynamic aspect of Ontology 4.0. Depending on the chosen level of thematization, the distinction between objects

and morphisms is flexible since a strategy of relativization determines the most appropriate construction.

Zambak (2014, p.72) states that

> The occurrence of mental activity in machine intelligence does mean a new kind of action of the highly dynamic representational system capable of making inferences from its experiences in order to achieve new results of action and form novel systems directed towards the future.

In light of this, we can say that Ontology 4.0 is not only dynamic but also evolving because each new contextual level analysis can ensure new trope compositions so that new semantic properties are involved in entities. In other words, unlike Ontology 3.0, where entities interact in a prescribed manner, Ontology 4.0 provides a framework in which structures in entities yield a meaningful network that can represent new trope compositions. Understanding the interactions between entities (which are urtrope compositions in a profound sense) and their exchanged semantics (which are preserved in categorial compositions; viz., in type theoretical approach) amount to merging dynamic facts from dispersed entities. That is the evolving feature of Ontology 4.0: it not only can re-structure the ontologies of contexts but also itself. It is because a part of the formalization of Ontology 4.0 is based on CT that models evolution and dynamic systems (Lawvere & Schanuel, 2009). For instance, natural transformations are used for ontology updates, or more generally, arrows expressed as dynamic inclusions are constituents of evolving and dynamic categories.

5.2.5 A theory of concurrency and interaction

Processing semantic properties is the gist of machine understandability, in which semantic prosperities are not limited to entity definitions of some domains. Instead, interactions among entities, or when we look more closely, interactions among semantic properties enlarge the possible inferences so that the machine can draw reliable conclusions. Moreover, the machine must compute concurrently since, for instance, in a context, several semantic properties and/or several types of relations can need to be processed, or in a factory, several machines

must communicate simultaneously. In this respect, interaction is required for handling open world issues, and showing that Ontology 4.0 is capable of representing any possible interaction is required. Thus, Ontology 4.0 must be an interactive system that allows concurrent computations. Thus, the marriage of features of being concurrent and interactive ends up with that Ontology 4.0 expresses interactive and concurrent computing.

We follow how Milner (1993) investigates modeling interaction and semantic basis of concurrent computation. He shows that entity-relation distinction does not work for modeling interactions so that λ-calculus is not the right tool for mathematizing interaction since its term-variable distinction offers modeling static structures. Even if λ-calculus is capable of representing any calculation, there must be another mathematical system that is capable of representing any interaction. To make the story short, Milner (1993, p.81) introduces π-calculus that is capable of representing any interaction: the key aspect of π-calculus is treating interactions and interactors, or agents, alike. Besides, he does not limit his investigation of the elements of interaction to entity-relation or program-memory interactions but rather extends it to any interactions at the discrete level; as such, the investigation includes real-life phenomena, such as the interaction with communication protocols and radio channels in a mobile telephone network.[3] In this respect, continual interactions include concurrency since it is quite unexceptional that an interactor interacts with different parties simultaneously. Thus, interaction and concurrency are taken as the constituents of a communication system, in which semantics play a crucial role. λ-calculus can handle concurrency with a functional structure but cannot deal with semantics and concurrency simultaneously. That is to say, modeling interactions and concurrency requires an investigation ın the semantic basis of interactive and concurrent computation. His investigations end up with the introduction of the Calculus for Communicating Systems.

[3] Of course, the mentioned mobile telephone technology goes back to the 1990s.

We will not discuss Milner's Calculus for Communicating Systems; instead, we use the features he found to show that Ontology 4.0 is a theory of interaction and concurrency by which various models can be developed. Urtrope theory treats interactions and interactors alike: everything is an urtrope composition. This feature enables the machine to process semantics concurrently. Moreover, category theory can represent π-calculus with functor categories[4] (Scott, 2000), that is to say, it is proven that there is a mathematical ground for Ontology 4.0 as a theory of interaction and concurrency. At this rate, representing everything with urtropes creates an exponential realm in which all the possible states reside. This space produces the capacity of both representing knowledge and processing that knowledge.

Last words on the interactive foundations of computing. Interactive systems are non-algorithmic, as they accept external inputs while they compute and express dynamic external behavior (Wegner, 1998). That is, dynamic and multiple inputs and outputs are key features of interactive systems. So, the machine must perform non-algorithmic or non-sequential computations if it is to handle works of ISW 4.0s. In principle, Ontology 4.0 solves this issue since it is formalized by category theory, which works with interactions rather than objects, as it is the structure language.

Several examples promise that CT meets the requirements for realizing the interactive foundations of computing. For instance, Goguen (1992) provides one of the earliest studies that explain phenomena in concurrent systems by utilizing sheaf theory—whose categorial abstraction is topos theory–and presents that *limit* can be the representation of behaviors of the individual objects and their interactions. Or consider, Abramsky, Gay, and Nagarajan (1997) deal with asynchronous concurrent systems by utilizing category theory. Furthermore, as a final note, a concurrent system needs not to be interactive; the same is true for an interactive system. However, the nature of ISW 4.0s includes all these two features. The machine can manage interactions and concurrency without omit-

[4] Note that it can also be represented with some other construction.

ting semantics, thanks to the urtrope and category theory. Thus, as Ontology 4.0 is a unified theory that underlines various models, the machine can handle interactions and concurrency.

5.2.6 An open system

Ontology 4.0 is constructed to deal with ISW 4.0s, which are open systems; besides, Ontology 4.0 is itself an open system. As we have explained being an open system at length, it suffices to state the following. Since all its components are in interaction, the semantic properties of entities are thought of as free variables. That is to say, a trope composition does not have a fixed type but rather is of a finite set of typed variables (Cf. Scott, 2000).

5.2.7 An interoperable system

Interoperability is "the ability of two or more software components to cooperate despite differences in language, interface, and execution platform" (Wegner, 1996, p. 285). Interoperability is the result of compatibility, which can be provided in many ways. For instance, bridge ontologies make distinct ontologies compatible; virtual machines provide interoperability of programming languages so that a system can utilize multiple languages at the same time, or interoperability APIs are created for integrating multiple data sources to be processed together. We purport that Ontology 4.0 provides interoperability in all these structures.

Any structured data representation, like databases, taxonomies, or ontologies, suffers from interoperability problems. These representation products are designed for specific problems, for specific domains, and/or from specific perspectives. As these are software products-not just an intellectual exercise, they are required to meet the needs of the companies/tasks/customers. Thus, integrating these representations means providing an overarching perspective, which is extra work.

Let us start with a database management problem: integrating databases. Sup-

pose that there are two distinct databases. In the first one, the names of employees are supposed to be entered as <first name|second name|surname>, whereas in the second one, they are supposed to be entered as <name|surname>. It is easy for humans to combine 'first name' and 'second name' columns and equalize them to the 'name' columns of the second database. However, what if shrinking two columns into one is not practical for some reason, how can the machine understand whether shrinking columns is a good idea and integrating distinct databases with correct values? Both questions are easy to answer for an expert in database management systems. Yet, these experts cannot answer how the column names/concepts that refer to some values are related to each other. That is an important question: How the column names are related to each other creates new categories that provide finding implicit relations. Well, the question addresses that the machine can process column names. The abovementioned example is a toy one; however, processing column names creates another level of investigation of interactions among column names and other values. Consequently, it is quite possible for the machine to draw a new column; that is, the machine discovers a new relation necessary for the database.

Ontologies also structure the data. In order to provide ontology interoperability, another ontology that glues others is required. That ontology can be a bridge ontology that is specifically designed for the ontologies that are supposed to be unified, or it can be a reference ontology that rules over the domain ontologies that can cohere, or it can be an upper-level ontology whose upper categories unite the entities of the domain ontologies. However, this approach is not economic: what should be done if two bridge, reference, or upper-level ontologies are asked to work harmoniously? For instance, how can Cyc and DOLCE work together, as they represent qualities from different perspectives? Then, there is a need for another ontology to integrate these ontologies. Thus, there must be a theory that automatically generates ontologies. Just as a Turing machine runs *any* algorithm, this theory can construct *an* ontology from *any* given data.

Data representations, however, are not limited to textual domains. Industry 4.0 requires interoperability among the devices that produce data, the data that has epistemic value. For instance, a device receiving signals from various

devices is to *understand* what each signal means and then performs accordingly. In the traffic, a signal says that many vehicles are accumulated at a crossroad; another says an elderly is just crossing the road, and another notices that an ambulance is coming through the crossroad. The machine that interprets all these signals has to decide what to do: turn the traffic light green to reduce traffic congestion; make all the vehicles wait till the ambulance passes by; wait till the elderly crosses the road; and then organize traffic lights. This kind of decision procedure cannot be modeled algorithmically since traffic is an open system. At this rate, signals have epistemic value, and the machine has to interpret them and perform accordingly. Lastly, note that the epistemic type of natural language and formal statements are different from the epistemic type of statements coming from signals. Nevertheless, different kinds of epistemic types must interoperate in order to realize machine-understandability. Thus, the representation of these epistemic types must be grounded on a machine ontology theory.

Last but not least, interoperability of different programs, either written in the same or different languages, is another critical trouble that is not specific to ISW 4.0s. It may require translations between the programming languages, models, and programs. Please note that this trouble also includes the integration of programming languages, which requires translations between models, data types, data structures, and even the formal system on which they are based. Semantics must be respected throughout these translations (Alagić & Bernstein, 2001). Note that semantics is not limited to databases or any kind of text-based document: it includes theories, models, and formal systems. Thus, the machine can also build ontologies for such things.

We claim that Ontology 4.0 provides interoperability in all these structures. Above all, interoperability requires standardization in representation. Urtrope theory standardizes data at the ontological level. As mentioned above, interactions and interactors are not differentiated in Ontology 4.0. Hence, once everything is represented as urtrope compositions, then interoperability occurs naturally. For instance, consider processing column names. As far as they are represented in urtrope/trope compositions, column names and the values under

those names can interact, just figuring out allowable morphisms. In other words, the machine can process not only the values of a column but also column names as values by figuring out the trope composition of the column name. Therefore, Ontology 4.0 changes the traditional database management approaches and offers a web of relations instead of relational data.

Secondly, type theoretical, in particular, homotopy type theoretical, basis of Ontology 4.0 contributes to interoperability since it does not concern representational differences. Recall that how the natural numbers are represented is not so important in a type theory: defining them as successive functions or binary encoding of numbers are just two distinct ways of encoding natural numbers. Type theory does not concern showing the differences between these encodings but rather ensuring that these different representations are equal. Using the Peano system instead of the binary system would make no difference from type theoretical perspective: when two things behave the same, they are considered the same. This idea is rooted in the Univalent Axiom of Voevodsky, which says "everything is preserved by equivalence" (Coquand, 2018).

The Univalent Axiom brings us to the third point: Voevodsky's notion of equivalence between types realizes that "all categorical constructions are preserved by isomorphism" and that "all constructions of categories are preserved by equivalence of categories" (Dybjer & Palmgren, 2020). Category theory, then, provides interoperability among programming languages, as it specifies the necessary conditions that determine equivalences between data types.

Considering type theories as a foundation for computation, Constable (1991) states that intuitional type theories are capable of representing programming logics, including dynamic logic. On the other hand, ontological accounts of some logical constants are inconstant; for instance, the ontological account of negation is changeable, so its implication and interpretation (Vickers, 2010). However, Ontology 4.0 provides standardization in logic as well. As a world representation is context-oriented, there is no monolithic representation of the world, no context-independent representations; each context is a different interpretation. Since category theory, topos theory, in particular, provides interoperability be-

tween different logics by providing a single mathematical structure (Alagić & Bernstein, 2001).

Consequently, Ontology 4.0 is capable of realizing that different representations are equal/the same and provides interoperability. Ontology 4.0 interoperates not only among the same structured systems, such as databases or programs, but also among different structured systems, such as databases and ontologies, or among protocols and programming languages. All the operations particular to interoperability are encapsulated in category theoretical computations, which the machine performs automatically. For instance, the machine investigates trope compositions of the columns and the values under the columns. Then, *automatically* creates (an) extra column(s) or shrinks the columns in the database. So, just as everything programmable can be translated into a Turing machine, different logical systems/software/databases/protocols, *in principle* can be translated into a common structure/platform that Ontology 4.0 provides.

5.2.8 A self-organizing system

A system is called self-organizing when it becomes structured by its own internal processes; in other words, it forms its structure without external control. Such a formation occurs from numerous interactions among components of the system (Yates et al., 1983), as such interaction paves the way for self-organization (Zambak & Vergauwen, 2007). Moreover, in a self-organizing system, internal interactions execute pattern formations without external interventions (Yates et al., 1983). For instance, the atmosphere is a self-organizational system in which weather conditions, and accordingly weather patterns, are changed in response to internal and/or external conditions, and such patterns emerge only within the system; that is, the process cannot be instructed from an outsider. At this rate, Ontology 4.0 is a self-organizing system: according to patterns illustrated by urtropes, decomposition and recomposition of urtropes execute pattern formations by ending up introducing new semantic properties to entities, generating new types, or creating hierarchies based on specific contexts. Furthermore, a pattern formation within Ontology 4.0 is not external; instead, it is inherent in

the morphisms and their typing rules. Thus, like in atmosphere, in Ontology 4.0, interactions —either directly or indirectly– among components give rise to emergent properties (Cf. Yates et al., 1983); that is, Ontology 4.0 is a system that can organize its structure over and over again without external intervention.

5.2.9 A generative system

It is discussed at length that Ontology 3.0 represents the world with webs of tags, the tags that determine the roles/behaviors of entities and relations with a context-based. The first thing to do is to list all the necessary entities of the domain and then list all the interactions among the entities determined by the purpose of constructing the ontology. We criticized this approach for being too labor-intensive and highly speculative in establishing a representation of the world as a knowledge web. That is, even if all the entities would be labeled in all the possible contexts, it is still impossible for machines to process the meanings through labels since labels that label the entities are also required to be labeled. Constructing a labeling system based on infinite regress is not smart at all.

The generative power of Ontology 4.0 stems from the urtrope theory: every representation is an urtrope composition. Unlike Ontology 3.0, there is no entity-relation dichotomy, which enables the machine to process everything on the same level. That provides efficacious interactions among urtropes and then tropes and trope compositions, and thanks to typing rules, the machine selects the allowable interactions from the web of morphisms. The allowable interactions that are novel to Ontology 4.0 are what the machine generates. In other words, the machine can automatically generate new tropes, new trope compositions, new types, and all the way to new ontologies according to a context. The trademark of Ontology 4.0 is that it can automatically figure out all the possible interactions of urtropes/tropes/entities/propositions/contexts, the interactions that are absolutely implicit to us. To sum up, Ontology 4.0 is a generative system that hinges on the fact that everything can be represented by urtropes, which are the non-extended building blocks of machine ontology.

5.2.10 A systematic system

Ontology 4.0 is generative; however, these generations are not throwaway: once a new type is detected, for instance, it is stored to be used in different settings. In this respect, Ontology 4.0 is systematic since it has the ability to apply its generations to other related structures. Let us unpack this feature.

Systematicity is a term that belongs to the philosophy of mind and computation used to explain how definite and predictable patterns are understood and used.[5] The key point of systematicity is to detect a composition's constituents and get a new composition without any additional information through the constituents of different compositions (Szabó, 2020). The machine draws the patterns by its ability to entertain among related structures. That is to say, the machine can relate the structures to entertain different compositions. The typical example of systematicity is that when one thinks that John loves Mary, they can entertain the thought that Mary loves John: "the capacity to think that John loves Mary is systematically related to the capacity to think that Mary loves John" (Rescorla, 2020). This situation can be customized to the machine. The machine can recombine the constituents freely and detect whether the combinations are sound, where the soundness depends on the composition rules. Because it can draw the structure of the given data and their semantic types; then it can figure out the possible interactions in the structure. Ontology 4.0 ensures that unstructured representations turn into structured representations where different potential compositions emerge (Cf. Schwitzgebel, 2021). Thus, the machine systematically entails that Mary loves John when it is given that John loves Mary.

[5] Our aim is not to discuss whether connectionism can explain systematicity or any debate on connectionism versus computationalism; instead, to show that Ontology 4.0 is systematic. Also, note that systematicity is discussed in the literature from a computational perspective, whereas, Ontology 4.0 promises systematicity from a non-computational perspective. Hence, the diverse approaches to systematicity are grounded on distinct ontological and computational perspectives.

5.2.11 A productive system

Productivity is another term borrowed from the philosophy of mind and computation, whose debates are ruled out from this work. We would rather take this term in the sense that a finite set of the primitive constituents and the rules for their combinations allows for building an infinity of constructions (Rescorla, 2020). In other words, productivity is about having the potential to create infinitely many wholes from finitely many parts. In light of this definition, it is legitimate to claim that Ontology 4.0 is productive. Here is why.

Please assume that the machine is given a complex statement[6] that it has never encountered before. If Ontology 4.0 is productive, then the machine *must understand* such a statement. As machine-understandability hinges on drawing semantic properties of entities constituting the statement, the machine can construct the sense of the statement out of semantic properties of entities by decomposing entities, namely the source, the target, and the relation, into trope compositions, figuring out the allowable compositions, and then choosing the most appropriate combination(s). That is, the machine can figure out the structure of the statement, which can be analyzed further into the structure of the constituents of the statement at the entity level and/or propositional level, and until the urtrope level analyses can be continued. Typing rules apply to draw the allowable morphisms till the machine guarantees the types of the statement and its components.

There are two final notes about the feature of productivity of Ontology 4.0. The first one is about the philosophical concern of unboundedness. If the machine is to process in a fair amount of time, the machine must have incredibility, high computation power, and infinitely large memory. However, cardinality is not a concern even for human intelligence, as natural languages are learnable and compositional at the same time (Szabó, 2020). This fact also holds for the organization that Ontology 4.0 provides. The machine can detect allowable

[6] It can either be a proposition or a context, as well.

morphisms from urtrope level to context level simultaneously. The other note is on figurative uses. Recall 'blue pen' and 'blue voyage' examples: even if both clauses have the same adjective, 'blue,' the sense it contributes differs. As explained in 'Figurative Meanings in Contexts' in section Contexts in CT, Ontology 4.0 provides a framework in which figurative meanings can be drawn thanks to its ability to detect structural correspondence at the trope level.

5.2.12 An associative system

One may wonder who first opened a wine bottle with a shoe and how these totally irrelevant entities –a shoe and a wine bottle – were associated. Letting aside the cognitive faculties of humans, in general, representing association is at the heart of discovery and innovation. From the machine perspective, we declared that associations are driven through the semantic properties, i.e., tropes and trope compositions. For instance, the machine highlights an association between smoking and lung cancer from causal interactions between the tropes in these entities. Note that proving associations is more complicated than finding patterns: associations explain the nature of patterns. The ability to associate semantic properties and morphisms with entities, relations, contexts and relations, relations and entities, facts and relations, and alike makes Ontology 4.0 the most powerful tool that finds associations.

5.2.13 A self-referential system

Previously, it was noted that self-reference is inevitable in typification, and relations must be self-reflexive. We use self-reference, self-representable, and self-reflexive interchangeably, considering they mean the ability of a system to represent itself in the same way it represents; the ability of a system to represent its constituents applying to themselves; and/or the ability of a system referring to itself. We planted the seed of Ontology 4.0 to be self-referential in part called "A Machine Ontology as a Self-Representable System" in section The Urtrope Theory, where we prosed urtrope theory as an untyped theory, which enables typification in an infinite number of times, without causing an infinite regress.

So it provides a vibrant representation that allows applying urtropes/tropes and their compositions to themselves. As such, Ontology 4.0 is self-referential, thanks to the urtrope theory that does the job without presenting paradoxes.

More precisely, Ontology 4.0 is self-referential since it is a system that can be represented by the same formal system that represents automatically generated context-specific ontologies. That is, as Ontology 4.0 can represent any ontology, it can also represent itself: there is no need for an upper ontology to represent Ontology 4.0. As urtropes bring together infinitely many forms, the system can represent itself innumerable by staying within its representational form. Hence, the urtrope theory, an ontology-based untyped theory, enables Ontology 4.0 to be turned on itself so that Ontology 4.0 can provide an account of its own existence (Cf. Kratzer, 2021).

5.2.14 A query-ready-structure provider

A query-ready structure is nothing but a knowledge web: entities of a context are linked to each other so that the machine can explore implicit links. In order to have this feature, first of all, the context must be structured. The ontologies generated by Ontology 4.0 have this feature. Ontology 4.0 operates on a context by structuring it; the context becomes query ready. In other words, a context is structured thanks to urtrope theory, and reference rules that are formalizations in category theory can be employed.

Moreover, Ontology 4.0 can model various types of reasoning, deductive, inductive, and case-based, to name some, thanks to CT. Ontology 4.0 also provides "a logically perspicuous representation of our commonsense understanding of the world as well as our scientific understanding" (Cocchiarella, 2007, p. xxiii).[7] Thus, Ontology 4.0 does automatically structure the data/create an ontology for the context: this is nothing but making the context ready for queries. In

[7] Cocchiarella (2007) lists criteria of adequacy for a formal ontology. Other list elements can be compared with the features of Ontology 4.0 in another study. We omit this work because Ontology 4.0 is not a formal ontology but a machine ontology.

other words, Ontology 4.0 can turn any unstructured data into a knowledge web specific to the context: it is for extracting the knowledge of the context.

5.2.15 An autopoietic system

We spoke of Ontology 4.0's being a self-organizing system: it can control its structural formation. On the other hand, we also spoke of Ontology 4.0's being an emergent system, as it can generate, for instance, new trope types. At this rate, we should consider whether introducing new tropes endangers the unity of Ontology 4.0 or, in the first place, why a system produces other components.

One of the essential paradigms of system theories is the dichotomy of system and environment. From a system theoretical perspective, systems are divided into open and closed ones. Closed systems have either no or limited interaction with their environments. On the other hand, open systems constantly interact with the environment they are in.[8] There is almost no surprise for emergence in a closed system but in an open system. Thus, systems have the property of emergence and produce new phenomena due to their interactions with the environments.

In the machine ontological parlance, a context is an environment, according to which solely Ontology 4.0 specifies the characteristics of the interactions. Given a context, the machine can generate an ontology specific to the context. This ontology specifies trope compositions and then can generate new compositions from those specifications that are unknown before the interaction. As Ontology 4.0 is a self-organizing system, it can form its own structure, but introducing a new component requires the system's formation. At this rate, we are not talking about a system that produces new compositions and eventually changes its own structure, but rather a system does furthermore attain new roles/behaviors to those compositions. Thus, Ontology 4.0 is more than being self-organizational.

[8] Please beware of the distinction that the operations in such a system are closed, yet its interactions are open.

In system theories, the feature that an open system can produce its own components and delegate their functions is called *autopoiesis*.[9] The new components find their characteristics and functions in the system thanks to the feature of self-organization. Namely, Ontology 4.0 can produce new components due to its interaction with contexts and then can assign new roles/functions to those components and form its structure by solely itself. In the words of Iba (2010): "an autopoietic system is defined as a unity whose organization is defined by a particular network of production processes of elements." Thus, the introduction of new tropes, for instance, cannot endanger the unity of Ontology 4.0. Said that the very feature of self-reference guarantees such a unity: the process of system formation requires the system being self-referential such that production of other components from the systems own components can be realized (Nöth, 2021). Alternatively, if Ontology 4.0 were not self-referential, it would assign the roles/behavior of new components under the command of some other system.

A citation from Varela (1979, p. 13) fits here well for illustrating an autopoietic system.

> *An autopoietic system is organized (defined as a unity) as a network of processes of production (transformation and destruction) of components that produces the components that: (1) through their interactions and transformations continuously regenerate and realize the network of processes (relations) that produced them; and (2) constitute it (the machine) as a concrete unity in the space in which they exist by specifying the topological domain of its realization as such a network.* [Emphasis in the original]
>
> It follows that an autopoietic machine continuously generates and specifies its own organization through its operation as a system of production of its own components, and does this in an endless turnover of components under conditions of continuous perturbations and compensation of perturbations. Therefore, an autopoietic machine[10] is a homeostatic (or rather a relations-static) system that has its own organization (defining network of relations) as the fundamental invariant. This is to be clearly understood. Every unity has an organization specifiable in terms of static or dynamic relations between elements, processes, or both. Among these possible cases, autopoietic machines are unities whose organization is defined by a particular network of processes (relations) of production of components, the autopoietic network, not by the components themselves or their static relations. Since the relations of production of components are given only as processes, if the processes stop the relations of production vanish; as a re-

[9] There are conceptual differences between system theories, so between the definition of autopoiesis, open system, self-organization, and so on. To see such differences, refer to Vidales and Brier (2021), especially Nöth (2021). The definitions are given here reflect our understanding.

[10] Beware that Varela (1979) uses *machine* to refer *living systems*.

> sult, for a machine to be autopoietic, its defining relations of production must be continuously regenerated by the components which they produce. Furthermore, the network of processes that constitute an autopoietic machine is a unitary system in the space of the components that it produces and that generate the network through their interactions.

To sum up, autopoiesis characterizes the organization of the systems that can produce their own components. To this respect, Ontology 4.0 is a unity whose organization is defined by category theoretical constructions, typing rules, and compositions of urtropes to form new components, a unity that determines the roles/behaviors of the components by itself. Additionally and lastly, thanks to this feature of Ontology 4.0, the machine does no longer suffer from the dichotomy of organization and structure; and this feature handles nonlinear/non-deterministic-dynamic contexts and the increasing complexity of the representations with the emergence of novel relations and types (Cf. Mainzer, 2004).

5.2.16 A Turing Machine of Ontologies

A *Turing machine* is a hypothetical machine that can simulate any computation. Today's computers can be taken as a Turing machine, as they can run various programs. For instance, word processors, video players, integrated development environments, and so on are run in a single machine without altering the hardware structure. Similarly, Ontology 4.0 is capable of automatically constructing an ontology that is particular to a context.

Ontology studies until Ontology 4.0 have focused on creating models that can fit contexts. For instance, RDF is a model used for structuring datasets, or an upper-level ontology is a model that is used for structuring domain ontologies. On the other hand, Ontology 4.0 is a system that introduces a structure that produces models. This is possible due that Ontology 4.0 is based on an untyped system that paves the way for typed systems. These typed systems occur in accordance with the context. In other words, each context has its model, viz., a domain ontology. Moreover, the structure of Ontology 4.0 can both represent and process itself; in other words, contentless abstract entities, urtropes, produce tropes endlessly, yet does it without enjoying an infinite regress.

Furthermore, inference systems until Ontology 4.0 have focused on creating models that can reflect the structure of the context with more expressibility and less complexity. For instance, description logics are used for the semantic web, or geometric logic is used for making inferences in moral agent systems. On the other hand, Ontology 4.0 is a system that introduces a mathematical framework that can generate different inference models. That is possible due that Ontology 4.0 is formalized in category theory, which can model different inference systems (Cf. Vickers, 2010). Thus, Ontology 4.0 is not only a Turing machine for representation but also for inference.

Before moving to the next feature, one caveat is necessary. Ontology 4.0 is not like a Turing machine in the sense that this hypothetical machine is extensional. Surprisingly, on the other hand, Ontology 4.0 is more like the λ-calculus for being non-extensional. It is by surprise since, most of the time, lambda calculus and Turing machines are considered the same model for computation. Additionally, neither lambda calculus nor Turing machines can express concurrency or interactions (Cf. Wegner & Goldin, 2003); but Ontology 4.0.[11]

5.2.17 Autonomy/Active agency

In the previous chapters, we have seen that all roads lead to autonomous machines for the realization of ISW 4.0s. Similarly, just as all roads lead to Rome, all the previous features of Ontology 4.0 form the path for the feature of autonomy. In this respect, the autonomy/active agency feature of Ontology 4.0 is the conclusion of all features.

Let us start with some definitions. Maes, cited in Zambak and Vergauwen (2007, p. 197), strongly defines agency:

> An *agent* is a system that tries to fulfill a set of goals in a complex, dynamic environment. An agent is situated in the environment: It can sense the environment through its sensors and act upon the environmental using its actuators. [...] An agent is called *autonomous* if it operates completely autonomously, that

[11] See the feature A theory of concurrency and interaction.

> is, if it decides itself how to relate its sensor data to motor commands in such a way that its goals are attended to successfully. [italics in the original]

In light of this definition, if Ontology 4.0 is an *autonomous agent*, or say *active agent*, then, for a given goal/situation, it should decide itself by structuring, analyzing, and interpreting a given context and then making inferences on the final product, while producing particular models for the context, not coercing to apply a fixed model. In other words, Ontology 4.0, as an agent, should determine appropriate semantic properties and process them within a collection of type rules according to the given context. However, Zambak (2014, p. 72) highlights the importance of accountability in the process of reaching the goals: "[t]he essence of agentive action is rationalization in which machine intelligence acts in order to achieve its goals." The goal can be defined as the processes of structuring, analyzing, interpreting, and making inferences in a context by processing the right semantic properties. Indeed, Ontology 4.0 has ontological, epistemological, and computational foundations that rationalize how it reaches its goals.

We will examine this feature, unlike the others, at length since it is the utmost important feature of Ontology 4.0. Before starting, let us note that autonomy and active agency are the same concepts for us. An agent can be responsive to its environment passively or actively. A passive response is mechanical or predetermined. An active response/act, on the other hand, can be beyond the conventional ones. For instance, opening a wine bottle with a shoe or using a full bottle of water as a doorstopper is a creative solution, not a usual one. With this respect, if the goal is to keep the door steady and if there is no doorstopper in the environment, an autonomous agent analyses the necessary and sufficient conditions within the environment that are useful for finding an object that substitutes/can be used as a doorstopper. So, a full bottle of water can substitute for a doorstopper by weight and volume. To sum up, an agent is active/autonomous when it can decide on things that can meet a given goal by utilizing things either in the system or in the environment out of their conditional situations. In other words, the agent can behave differently from a particular behavior. That also means creativity is of autonomous systems since discovery

is a product of the system itself. As Iba (2010) defines, a sequence of discoveries can be taken as a creative process. Explicitly, with respect to the system itself or the environment, the system discovers within a collection of its operations, the operations performed to reach the goal. Consequently, such a system is both operationally closed and creative.

Now it is time to show how Ontology 4.0 behave autonomously, albeit prescribing certain laws and rules, and how it can give an account of its actions. For this illustration, we will follow the *heteromorphic theory of adjunction*, developed by Ellerman (2006). According to this theory, autonomous behavior can emerge within a system that has to follow certain laws, and further, such behaviors are determined through universals (Ellerman, 2006, p.174):

> In grand philosophical terms, the factorization through universals of an adjunction gives an approach, albeit in rather abstract mathematical terms, to resolving what is perhaps the central conundrum of philosophy, the reconciliation of external determination ("necessity" or "heteronomy") and self-determination ("freedom" or "autonomy").

Ellerman (2006) applies his theory in empirical and social sciences.[12] As we will see, this theory already applies to Ontology 4.0. Now let us appreciate the theory, its examples, and its implementation in Ontology 4.0 in turn.

5.2.17.1 The heteromorphic theory about adjunction

Ellerman's idea of "determination though universals" can be reduced to the idea of function –for the sake of demonstration-: "a function describes how one thing determines another" (p. 20). For instance, given two sets of integers, X and Y, and a function $f : X \longrightarrow Y$, f assigns each element of X to some element of Y. The elements of X are the determiners, and the elements of Y are the potential determinees. They are called "potential" because $f(x)$ is the actual determinee, where $x \in X$ and $f(x) \in Y$. Then, given $f : \mathbb{Z} \longrightarrow \mathbb{Z}$ and $f(x) = 2x$, 2 determines that the outcome is 4, though the determination of $2x$. That is, the source is the determiner –sender–, the target is the determinee –receiver–, and

[12] He also uses this theory as a basis for social engineering. See Ellerman (2009) for more detail.

the function is the determination. In category theoretical parlance, the determiners and the determinees are objects in categories, and the determinations are the morphisms between objects. However, the morphisms are of different types when the objects are of different categories. We are familiar with the use of *homomorphisms*, as we have mentioned that they are morphisms in a category, namely, arrows. When objects are of different categories, they are no more arrows but *heteromorphisms*. For instance, a morphism from an object of `Mon` to an object of `Aut` is a heteromorphism. Thus, we can speak of a heteromorphic determination just as a homomorphic one.

Recall that universals are essential notions, just like natural transformations. Their invaluable status provides, for instance, that they transfer knowledge between objects satisfying the same universal in different theories or allow studying an object from various perspectives. Ellerman proposes that the construction of a universal paves the way for autonomy. Said that, is it unquestionable that the source or the target can be universally represented? In order to be sure about the mathematical constructions of any example, we need to work with adjoint functors. In the part Adjoints in chapter An Orchestration of the Urtrope and Category Theories, we noted that adjoint functors could unify and/or subsume categorial construction; many mathematical theorems can be rewritten as statements about the existence of adjoint functors; they adhere to syntax and semantics, and so on. They are primarily in this study because every universal constructor can be defined in terms of adjoints. Ellerman's theory is based on this fact about adjoints.

More precisely, adjoint functors take a "given object to its corresponding universal object in the other category" (Ellerman, 2007, p. 24). Therefore, the focus should be on an adjunction,[13] which arises from (1) where every heteromorphism to a given object in the target category can be universally represented within the source category, and 2) where every heteromorphism from a given object in the source category can be universally represented within the target category

[13] Recall that an adjunction is a pair of adjoint functors, namely left and right adjoints.

(Ellerman, 2007, p. 24). The universal models are successful if a homomorphism can represent each heteromorphism as an internal map. So, autonomy results from such a construction: the universality + the internality. Ellerman's theory is summarised in Figure 5.1:[14]

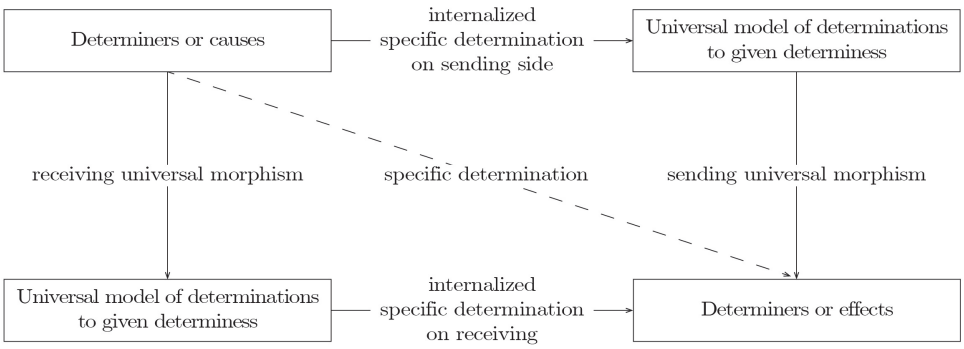


Figure 5.1: Adjunctive square as general scheme for determination through universals

This square explains the theory. Let us unpack this square with the following illustrations and move on to the examples where we will examine this square from various angles to understand the mathematical theory of autonomy.

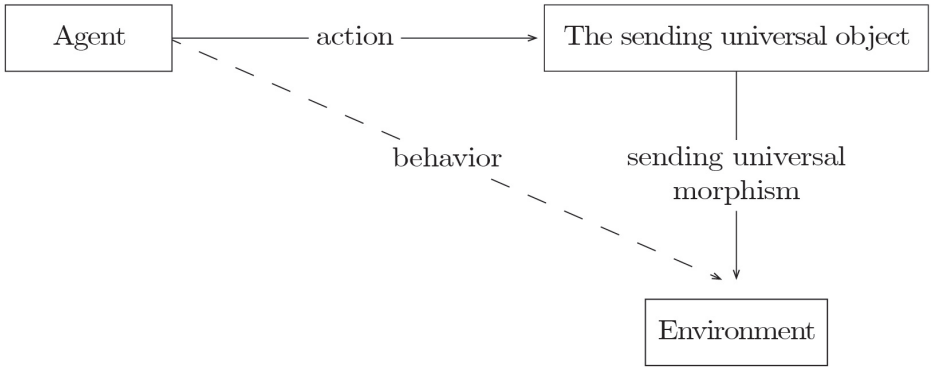
5.2.17.2 Examples of Ellerman's Theory

Internalization of a specific behaviour as an action : Suppose that there is an agent who is in an environment, and the agent somehow affects the environment. This direct observation is the behavior of the agent in the environment. In category theoretical parlance, certain behavior is a morphism from the agent to the environment. That is a direct determination; however, claims Ellerman (2007), the agent needs to internally construct a representation of all possible behaviors in order to figure out indirect determinations if it is autonomous. That is, the construction of a corresponding universal internalizes the mentioned behavior as an action, among other possible behaviors. The notion of *internalization* refers to the fact that the agent and the corresponding universal are of the same type. The effect of the behavior in the environment

[14] The original figure is Figure 5 on page 27.

can be reached through the sending universal object, by the sending universal morphism. Here, the construction of the universal constructs the collection of all effects so that the given instance of behavior factors through the universal by the internal morphism, viz., behavior. This internal morphism "chooses" the effects and sends the same result to the environment as the original interaction from the agent.

Please note two things: (1) it is not the agent by itself the "sender," rather, the agent and the universal object together is the sender. So, in this example, the receiver is the environment alone. (2) "determination through a universal" is not solely about the universal object but includes the universal morphisms. For instance, the universal of product cannot be thought of without the projections. Thus, for this example, "the sending universe" combines the sending universal object and the sending universal morphism.

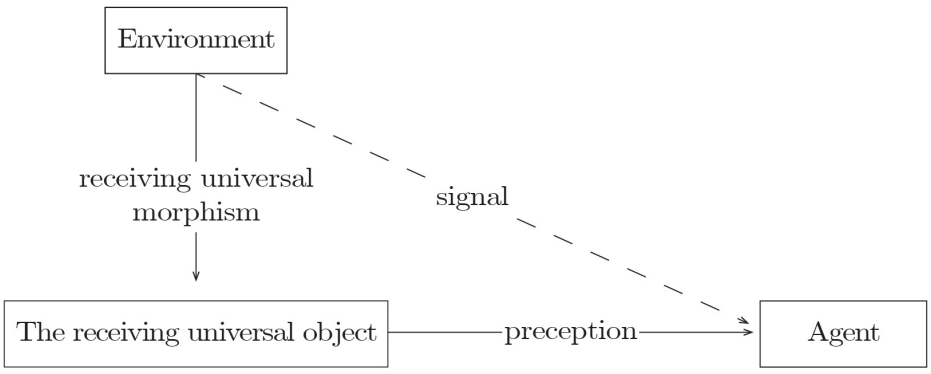


The diagram depicts that the specific behavior can be factored through the universal by the specific internal behavior. In other words, the observed behavior can be determined through the internal representations of the agent's behaviors. Ellerman (2007) shows that mathematically the observed behavior and the internal factorization through the universal give the same result; consequently, the above diagram commutes.

An internal determination through a universal is worth examining more than the direct external determinations because only the former ends up with autonomy, says Ellerman (2007). He states that "[t]he net effect is that the sender ("organism") is "disconnected" from direct "causal" interaction with the effects

(the action takes place, as it were, in the internalized "world") and becomes in that sense autonomous" (p. 29). Namely, the agent and the environment are "separated," such that the given behavior, like other actions, occurs in the "exemplary" environment internalized by the agent.

Internalization of a signal as perception : Suppose that there is another agent who is in an environment. In this example, the sender is the environment, the receiver is the agent, and the environment sends signals to the agent. In category theoretical parlance, a certain signal is a morphism from the environment to the agent. This is a direct determination; however, claims Ellerman (2007), the agent needs to internally construct a representation of all possible signals from the environment in order to figure out indirect determinations. That is, the construction of a corresponding universal internalizes the mentioned signal as perception. The notion of *internalization* here also refers to the agent and the corresponding universal being of the same type. The signal affects the agent and can be reached through the receiving universal object by the receiving universal morphism. The universal construction, here, constructs the collection of all signals so that the given instance of signal factors through the universal by the internal morphism, viz., perception. This internal morphism "perceives" the signal and sends the same signal to the agent as the original interaction from the environment. Again note that (1) it is not the agent by itself the "receiver," but rather, the agent and the universal object together. So, in this example, the sender is the environment alone. (2) Determination through a universal is not solely about the universal object but includes the universal morphisms. For instance, the universal of coproduct cannot be thought of without the injections. Thus, for this example, "the receiving universe" combines the receiving universal object and the receiving universal morphism.



The diagram depicts that the specific signal can be factored through the universal by the specific internal perception. In other words, the detected signal can be determined through the internal representations of signals of the agent. Ellerman (2007) shows that mathematically the internal perception of the signal through the universal is the same as the detected signal; consequently, the above diagram commutes.

Ellerman (2007, p. 26) states that "[i]n more philosophical terms, the internalized determination through the receiving universal gives the receiving "organism" [agent] a certain measure of independence or autonomy from the direct stimulus control represented by the specific external determinations." In this respect, the receiving agent can construct a "separate internal environment" that provides a kind of autonomy for its environment, thanks to the ability to construct universals and their internalizations.

Universal Turing machine as a sending universal object : Ellerman (2007, p. 30) offers a representation in which a universal Turing machine is a universal object, such that a special-purpose Turing machine performs only a specific calculation and whose result is equal to the result which is produced by factorization through a universal Turing machine – as the universal object. That is to say, a special-purpose TM directly produces certain results for the given input; in contrast, the same result can be reached by internal determination through a universal Turing machine. So, the following diagram, Figure 5.2, commutes.

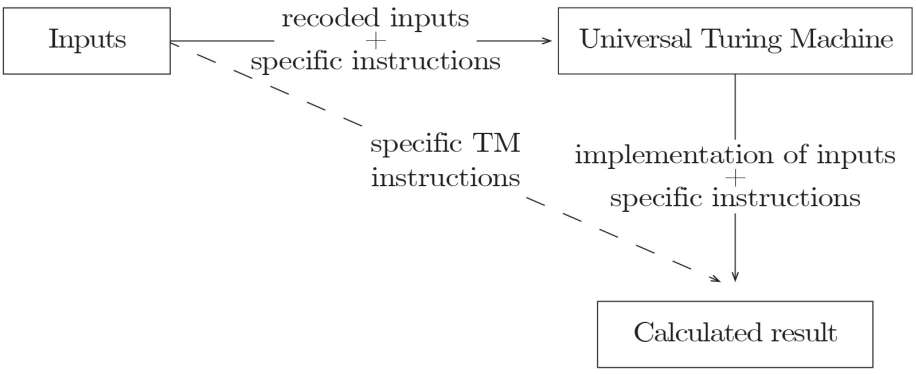


Figure 5.2: Special-purpose calculator factored through a universal TM

Here the autonomy comes from the idea that the input can be an object to various TM instructions, all of which can be represented in a universal TM. Or in other words, the inputs are independent of specific instructions of a special-purpose TM.

Universal language faculty as a receiving universal object : Ellerman (2007, p. 36) illustrates Chomsky's theory of generative grammar by employing the adjunction square. Chomsky's theory of generative grammar claims that, in general terms, a child does not learn grammar rules of the mother's tongue, but the linguistic experience of the child "selects" the rules (ibid.). Figure 5.3 is the commuting diagram of Chomsky's grammar.

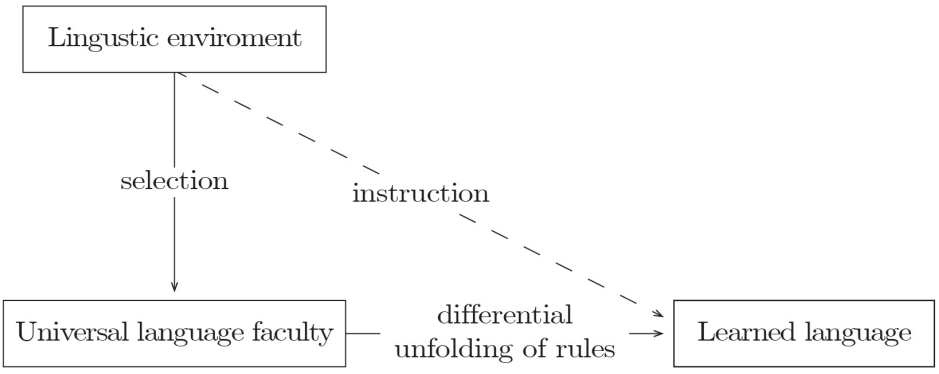


Figure 5.3: Generative grammar account of language learning as determination through a universal

This is an example of indirect determination: we understand *any* natural language through our universal language faculty, and we experience a language

sufficiently. Moreover, the universality of the internal mechanism offers "the limitless possibilities of thought and imagination" that are "reflected in the creative aspect of language use" (Chomsky (1966), in Ellerman (2007, p. 37)) Thus, the autonomy is rooted in the universal language.

5.2.17.3 Implementation of Ellerman's Theory in Ontology 4.0

We purport that the heteromorphic theory of adjunction suggests a conceptual structure of how autonomy can also emerge within Ontology 4.0. The urtrope theory claims that urtropes compose endlessly and creates an infinite number of properties, viz., tropes. The proper construction and meaningful selection of these tropes and trope compositions require evaluating them in interaction. That is also a result of tropes and their compositions – entities gaining properties, contexts—gaining their meaning in interaction. So, the heteromorphic theory of adjoints will help construct and demonstrate the machine's autonomy since tropes and everything deriving from their compositions can be described in a topos theoretical structure; because all ontological structures that we represent as toposes are built on universals such as an exponential object, terminal object, product, and all of them can be represented with adjoint functors. Thus, choosing the right adjunction for Ontology 4.0 ensures its autonomous structure.

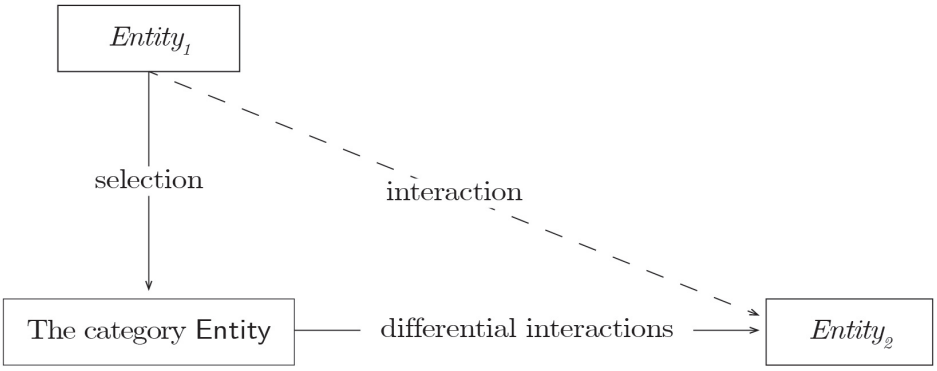


Figure 5.4: Choosing the right morphism between entities through `Entity`

An entity by itself has no meaning at all; nevertheless, it gains its meaning by interacting with other entities in a context. *Gaining meaning* means that the ontological status of the entity becomes explicit. Besides, the entity interacts

with others through some of its semantic properties. An interaction occurs between the entity's semantic property and another entity's semantic property in the context. For instance, in the proposition "Laura jumps," a semantic property of 'Laura' and a semantic property of 'jumping' are in interaction, and the existence of a morphism between these proves the proposition.

Moreover, recall that semantic properties are prior to entities: interactions among semantic properties determine the entities since Ontology 4.0 adopted relation-based representation. Following these, we can portray that the determination, namely a specific interaction, is heteromorphic when the interaction is taken to be between two entities. That is to say, an object, a semantic property, of an entity, $Entity_1$, is linked to an object, a semantic property, of another entity, $Entity_2$. As can be seen in Figure 5.4, the determination is the link between the semantic properties of two distinct entities. However, the machine needs to select the appropriate morphism between the entities. The category `Entity` of entities, whose objects are entities and arrows are allowable interactions, is a receiving universal object through which internalization can happen. In other words, `Entity` is the universal through which the morphism between $Entity_1$ and $Entity_2$ can be determined. The category `Entity` of entities is the generator of the multiplicity in the sense that it exhibits all the interactions of $Entity_2$. Thus, the internalization in the receiving side offers a functor, call it differential interactions, which maps the appropriate entity to $Entity_2$. Consequently, determination through universals provides autonomy so that the machine can determine the morphism between the entities that are particular to the semantic properties of entities.

Suppose that all the types of arrows in a context are determined. Behind the determination of the types of arrows between entities, there are certain semantic properties of the entities. We expect the machine to act according to the type rules and make certain inferences. However, entities also have other semantic properties. The machine must decide which of these to use in that context. This process means processing the background information. Figure 5.5 illustrates a specific interaction internalized through a universal sending object. This interaction determines what other semantic properties of an entity makes the entity

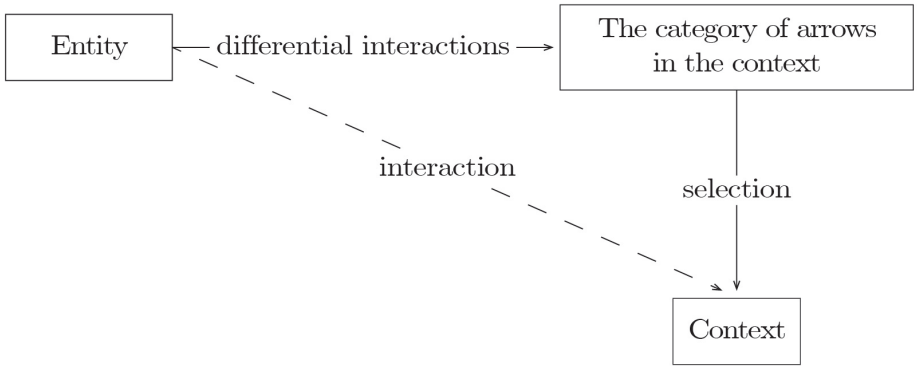


Figure 5.5: Emerging implicit semantic properties of an entity in a context

interact with other entities in a context. The determination is heteromorphic since the objects, the semantic properties, of the chosen entity interact with entities in the context where the entity is. In this scenario, the machine needs to attain morphisms from ⌜ENTITY⌝ to other entities in ⌜CONTEXT⌝. We will define a category whose objects are the arrows of the ⌜CONTEXT⌝, and the arrows are the typing rules so that all the contextual interactions are defined in a category.[15] With the application of typing rules to all the objects, a representation of all possible interactions in the context is obtained. Thus, the differential interactions from ⌜ENTITY⌝ to the category of all contextual interactions provide a list of all the possible interactions between the entity and the other entities in the context. However, not all are important for the context, so the machine selects the appropriate ones that emerge the necessary background information. Consequently, determination through a universal provides autonomy so that the machine can select the morphisms that can emerge in a context, the morphisms that are hidden in the trope compositions in entities.

5.2.17.4 Conclusion

The idea of determination through universals, mathematically express adjunctions, unveils various collections of possibilities, which are indirect determina-

[15] This category is considered like a functor category. Constructing such a category is legitimate since all the way, we are working with Cartesian closed categories.

tions. Therefore, some measure of autonomy can exist when there are indirect factorizations through universals. In other words, determination-through-universals puts forward determinations other than a direct external determination.

Ontology 4.0 has finite typing rules, and there is an infinite possibility of semantic types due to urtrope compositions. The limitless possibilities of urtrope compositions are reflected in the creative aspect of semantic type generation. However, limitless possibilities of urtropes cannot be observed directly, just like the possible interactions might occur in a context. In a context, external determinations are observable by nature; however, the most crucial potency of Ontology 4.0 lies in the ability to discover implicit interactions among semantic properties. In either case, an interaction can be restructured as a determination through an internal universal so that a universal model of determinations can present qualitatively different types of interactions. In Ellerman (2007, p. 38)'s words, "[t]he internalization through the universal structure builds a "separate" internalized "space" or "world" and thus supports the emergence of a qualitatively new level of relatively autonomous activity that would not otherwise be present if there was only the direct determinative connections." Consequently, Ontology 4.0 has the feature of autonomy/active agency as it can reach its goals in the process of structuring, analyzing, interpreting, and making inferences through constructing adjunctions.

5.3 Future Works

The final words of this work are about possible future works. The very first future work concerns representing entities in terms of urtropes. Indeed, we have started a project aiming to classify Web contents upon a search. We are employing an urtrope theoretical approach to constructing tropes.

Once an urtrope theoretical representation is constructed, we will formalize them in topos theory. Applied category theory studies and natural language processing techniques are the essential components of this work. Our model will be tested

against state-of-the-art models for its complexity, efficacy, and inference power.

Another future work is to elaborate on the features of Ontology 4.0. Although this work explains how Ontology 4.0 has such features, it is also necessary to formalize them, just like the construction of a Turing Test or building models as Milner (1993) did with π-calculus to model interaction between machines.

Lastly, this work can be taken as a comment on metaphysics. Current philosophical attitudes can be questions from a machine-understandability perspective. For instance, there can be other perspectives to admit and define abstraction.

BIBLIOGRAPHY

Abbott, M. R. (2009). A new path for science? *The Fourth Paradigm: Data-Intensive Scientific Discovery*, 111–116.

Abramsky, S., Gay, S., & Nagarajan, R. (1997). A type-theoretic approach to deadlock-freedom of asynchronous systems. In *International symposium on theoretical aspects of computer software* (pp. 295–320).

Ackoff, R. L. (1989). From data to wisdom. *Journal of Applied Systems Analysis*, *16*(1), 3–9.

Adam-Bourdarios, C., Cowan, G., Germain, C., Guyon, I., Kégl, B., & Rousseau, D. (2015). The Higgs boson machine learning challenge. In *Nips 2014 workshop on high-energy physics and machine learning* (pp. 19–55).

Adámek, J., Herrlich, H., & Strecker, G. E. (2004). *Abstract and concrete categories: The joy of cats*. Citeseer.

Adriaans, P. (2013). Information. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Fall 2013 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/fall2013/entries/information`.

Alagić, S., & Bernstein, P. A. (2001). A model theory for generic schema management. In *International workshop on database programming languages* (pp. 228–246).

Allemang, D., & Hendler, J. (2011). *Semantic web for the working ontologist: effective modeling in rdfs and owl*. Elsevier.

Alvarado, J. T. (2019). Are tropes simple? *Teorema: Revista Internacional de Filosofía*, *38*(2), 51–72.

Analytics, I. (2015). *Big data analytics.* Retrieved January 23, 2018, from `https://www.ibm.com/analytics/hadoop/big-data-analytics`.

Andersen, H., & Hepburn, B. (2016). Scientific method. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Summer 2016 ed.). Meta-

physics Research Lab, Stanford University. https://plato.stanford.edu/archives/sum2016/entries/scientific-method.

Anderson, C. (2008). The end of theory: The data deluge makes the scientific method obsolete. *Wired Magazine*, *16*(7).

Arp, R., Smith, B., & Spear, A. D. (2015). *Building ontologies with basic formal ontology*. MIT Press.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., ... others (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, *58*, 82–115.

Asperti, A., & Longo, G. (1991). *Categories, types, and structures: an introduction to category theory for the working computer scientist*. MIT Press.

Auslander, M., & Buchsbaum, D. (1974). *Groups, rings, modules*. Courier Corporation.

Awodey, S. (2006). *Category theory*. Oxford University Press.

Baclawski, K., Bennett, M., Besrg-Cross, G., Fritzsche, D., Sharma, R., Singer, J., ... Whitten, D. (2020). Ontology summit 2019 communiqué: Explanations. *Applied Ontology*, *15*(1), 91–107.

Baez, J. (2021). *Topos theory in a nutshell*. Retrieved January 03, 2022, from https://math.ucr.edu/home/baez/topos.html.

Barnes, J., et al. (1995). *Complete works of Aristotle: The revised Oxford translation* (Vols. 1–2). Princeton University Press.

Barr, M., & Wells, C. (1998). *Category theory for computing science* (Third Edition ed.). https://www.math.mcgill.ca/triples/Barr-Wells-ctcs.pdf.

Barroso, C. A. C. (2014). A ontologia tractatiana. *Síntese: Revista de Filosofia*, *41*(130), 217–238.

Bealer, G. (1994). Property theory: The type-free approach v. the Church approach. *Journal of Philosophical Logic*, *23*(2), 139–171.

Bealer, G., & Mönnich, U. (2003). Property theories. In *Handbook of philosophical logic* (pp. 143–248). Springer.

Beaney, M. (2013). The Oxford handbook of the history of analytic philosophy. In M. Beaney (Ed.), (pp. 3–29). Oxford University Press.

Beaney, M. (2018). Analysis. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Summer 2018 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/sum2018/entries/analysis`.

Bell, J. L. (1981). Category theory and the foundations of mathematics. *The British Journal for the Philosophy of Science*, *32*(4), 349–358.

Bench-Capon, T., & Malcolm, G. (1999). Formalising ontologies and their relations. In *International conference on database and expert systems applications* (pp. 250–259).

Bender, E. M., & Koller, A. (2020). Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 5185–5198).

Benzmüller, C., & Andrews, P. (2019). Church's Type Theory. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Summer 2019 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/sum2019/entries/type-theory-church`.

Berners-Lee, T. (1989). *Information management: A proposal* (Tech. Rep.).

Berners-Lee, T., Fielding, R., & Masinter, L. (2005). *Uniform resource identifier (URI): Generic syntax.* Retrieved March 03, 2018, from `http://www.ietf.org/rfc/rfc3986.txt`.

Berners-Lee, T., & Fischetti, M. (2001). *Weaving the web: The original design and ultimate destiny of the world wide web by its inventor.* Diane Publishing Company.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American*, *284*(5), 34–43.

Berners-Lee, T., & Swick, R. (2006). *Semantic web development* (Tech. Rep.). Massachusetts Institute of Technology, Cambridge.

BFO Contributors. (2017). *Basic formal ontology.* Retrieved June 23, 2018, from `http://basic-formal-ontology.org`.

Biran, O., & Cotton, C. (2017). Explanation and justification in machine learning: A survey. In *IJCAI-17 workshop on explainable AI (XAI)* (Vol. 8, pp. 8–13).

Biss, D. K. (2003). Which functor is the projective line? *The American Mathematical Monthly*, *110*(7), 574–592.

Blass, A. (1984). The interaction between category theory and set theory. *Contemporary Mathematics*, *30*, 5–29.

Boden, M. A. (2016). *AI: Its nature and future*. Oxford University Press.

Bogen, J. (2017). Theory and observation in science. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Summer 2017 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/sum2017/entries/science-theory-observation`.

boyd, d., & Crawford, K. (2012). Critical questions for big data: Provocations for a cultural, technological, and scholarly phenomenon. *Information, Communication & Society*, *15*(5), 662–679.

Bradley, F. H. ([1893] 2016). *Appearance and reality: a metaphysical essay*. Routledge.

Bradley, T.-D. (2018). What is applied category theory? *arXiv preprint arXiv:1809.05923*.

Brain Simulation Platform. (2017). Human Brain Project. Retrieved July 11, 2019, from `https://www.humanbrainproject.eu/en/brain-simulation/brain-simulation-platform`.

Brentano, F. (1975). *On the several senses of being in Aristotle* (R. George, Trans.). University of California Press.

Bricker, P. (2016). Ontological commitment. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Winter 2016 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/win2016/entries/ontological-commitment`.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... others (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Buchanan, R. A. (2018). *History of technology.* Encyclopædia Britan-

nica, Inc. Retrieved June 09, 2018, from https://www.britannica.com/technology/history-of-technology.

Bush, V. ([1945] 1979). As we may think. *ACM Sigpc Notes*, *1*(4), 36–44.

Cafezeiro, I., & Haeusler, E. H. (2007). Semantic interoperability via category theory. In *ER (tutorials, posters, panels & industrial contributions)* (pp. 197–202).

Cafezeiro, I., Haeusler, E. H., & Rademaker, A. (2008). Ontology and context. In *2008 Sixth annual IEEE international conference on pervasive computing and communications (percom)* (pp. 417–422).

Cafezeiro, I., Viterbo, J., Rademaker, A., Haeusler, E. H., & Endler, M. (2014). Specifying ubiquitous systems through the algebra of contextualized ontologies. *The Knowledge Engineering Review*, *29*(2), 171–185.

Campbell, K. (1990). *Abstract particulars.* Oxford Basil Blackwell.

Cao, L. (2017). Data science: Challenges and directions. *Communications of the ACM*, *60*(8), 59–68.

Caramello, O. (n.d.). *Teaching - Olivia Caramello's website.* Retrieved from https://www.oliviacaramello.com/Teaching/Teaching.htm. (The videos and slides of lectures have been studied several different times.)

Caramello, O. (2018). *Theories, sites, toposes: Relating and studying mathematical theories through topos-theoretic 'bridges'.* Oxford University Press.

Cardelli, L., & Wegner, P. (1985). On understanding types, data abstraction, and polymorphism. *ACM Computing Surveys (CSUR)*, *17*(4), 471–523.

Çevik, A. (2019). *Matematik felsefesi ve matematiksel mantık.* Nesin Yayıncılık.

CERN. (2013). *The birth of the Web.* Retrieved September 08, 2018, from https://home.cern/topics/birth-web.

CERN. (2019). *The Higgs Boson.* Retrieved July 11, 2019, from https://home.cern/science/physics/higgs-boson.

Chan, C. K., Lee, Y. C., & Lin, V. (2009). Harnessing Web 2.0 for collaborative learning. In *13th international conference on biomedical engineering* (pp. 2171–2172).

Cocchiarella, N. B. (2007). *Formal ontology and conceptual realism.* Springer.

Coecke, B., Sadrzadeh, M., & Clark, S. (2010). Mathematical foundations for a compositional distributional model of meaning. *arXiv preprint arXiv:1003.4394*.

Colomb, R. M., Dampney, C. N., & Johnson, M. (2001). Category-theoretic fibration as an abstraction mechanism in information systems. *Acta Informatica*, *38*(1), 1–44.

Constable, R. L. (1991). Type theory as a foundation for computer science. In *International symposium on theoretical aspects of computer software* (pp. 226–243).

Copeland, B. J. (2017). The modern history of computing. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Winter 2017 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/win2017/entries/computing-history`.

Coquand, T. (2018). Type Theory. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Fall 2018 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/fall2018/entries/type-theory/`.

Cormode, G., & Krishnamurthy, B. (2008). Key differences between Web 1.0 and Web 2.0. *First Monday*, *13*(6).

Crupi, V. (2016). Confirmation. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Winter 2016 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/win2016/entries/confirmation`.

Cruse, D. A. (2017). The lexicon. In *The handbook of linguistics* (p. 235-254). John Wiley and Sons. doi: https://doi.org/10.1002/9781119072256.ch12

Cycorp. (2018). *Cycorp.* Retrieved December 28, 2018, from `https://www.cyc.com`.

Davis, M. (2000). *Engines of logic: Mathematicians and the origin of the computer* (Vol. 7). Norton New York.

Davis, R., Shrobe, H., & Szolovits, P. (1993). What is a knowledge representation? *AI magazine*, *14*(1), 17.

Demey, L., Kooi, B., & Sack, J. (2019). Logic and probability. In E. N. Zalta

(Ed.), *The Stanford encyclopedia of philosophy* (Summer 2019 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/sum2019/entries/logic-probability`.

Domingos, P. (2015). *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books.

Doran, D., Schulz, S., & Besold, T. R. (2017). What does explainable AI really mean? A new conceptualization of perspectives. *arXiv preprint arXiv:1710.00794*.

Došen, K. (2003). Identity of proofs based on normalization and generality. *Bulletin of Symbolic Logic*, *9*(4), 477–503.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Dybjer, P., & Palmgren, E. (2020). Intuitionistic Type Theory. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Summer 2020 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/sum2020/entries/type-theory-intuitionistic`.

Dyche, J. (2012). Big data 'eurekas!' don't just happen. *Harvard Business Review Blog*, *20*.

Effingham, N. (2013). *An introduction to ontology*. John Wiley & Sons.

Ehrich, H.-D., Goguen, J. A., & Sernadas, A. (1990). A categorial theory of objects as observed processes. In *Workshop/school/symposium of the rex project (research and education in concurrent systems)* (pp. 203–228).

Ehring, D. (2011). *Tropes: Properties, objects, and mental causation*. OUP Oxford.

Ellerman, D. (2006). A theory of adjoint functors with some thoughts on their philosophical significance. In *What is category theory?* (Vol. 3, p. 127).

Ellerman, D. (2007). Adjoints and emergence: applications of a new theory of adjoint functors. *Axiomathes*, *17*(1), 19–39.

Ellerman, D. (2009). *Helping people help themselves: From the world bank to an alternative philosophy of development assistance*. University of Michigan Press.

Encyclopædia Britannica. (n.d.). *Data processing*. Encyclopædia Britannica,

Inc. Retrieved March 02, 2020, from https://www.britannica.com/technology/data-processing.

Encyclopædia Britannica. (2011). *Industry.* Encyclopædia Britannica, Inc. Retrieved September 08, 2018, from https://www.britannica.com/technology/industry.

Falguera, J. L., Martínez-Vidal, C., & Rosen, G. (2022). Abstract Objects. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Summer 2022 ed.). Metaphysics Research Lab, Stanford University. https://plato.stanford.edu/archives/sum2022/entries/abstract-objects.

Feferman, S. (1977). Categorical foundations and foundations of category theory. In *Logic, foundations of mathematics, and computability theory* (pp. 149–169). Springer.

Fermilab and the Higgs Boson. (2019). Fermi Research Alliance, LLC. Retrieved January 26, 2021, from https://www.fnal.gov/pub/science/higgs/tevatron.html.

Findlay, J. (1936). Relational properties. *The Australasian Journal of Psychology and Philosophy*, *14*(3), 176–190.

Fitting, M. (2020). Intensional Logic. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Spring 2020 ed.). Metaphysics Research Lab, Stanford University. https://plato.stanford.edu/archives/spr2020/entries/logic-intensional.

Floridi, L. (2011). *The philosophy of information.* Oxford University Press.

Floridi, L. (2012). Big data and their epistemological challenge. *Philosophy & Technology*, *25*(4), 435–437.

Føllesdal, D. (1996). Analytic philosophy: what is it and why should one engage in it? *Ratio*, *9*(3), 193–208.

Fong, B., & Spivak, D. I. (2018). Seven sketches in compositionality: An invitation to applied category theory. *arXiv preprint arXiv:1803.05316*.

Ford, L. S. (1983). *Explorations in whitehead's philosophy.* Fordham University Press.

Foskett, D. J., Estabrook, L. S., Francis, F. C., & Haider, S. (n.d.). *Library.* Encyclopædia Britannica, Inc. Retrieved February 03, 2020, from https://www.britannica.com/topic/library.

Freeman, C., & Louça, F. (2001). *As time goes by: the information revolution and the industrial revolutions in historical perspective.* Oxford University Press.

Frické, M. (2015). Big data and its epistemology. *Journal of the Association for Information Science and Technology*, *66*(4), 651–661.

Fromkin, V., Rodman, R., & Hyams, N. (2018). *An introduction to language* (11th ed.). Cengage Learning.

Galmiche, D. (1990). Constructive system for automatic program synthesis. *Theoretical Computer Science*, *71*(2), 227–239.

Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, *35*(2), 137–144.

Gartner—IT. (2013). *Big data defintion.* Retrieved January 23, 2018, from `https://www.gartner.com/it-glossary/big-data`.

Gitelman, L. (2013). *Raw data is an oxymoron.* MIT Press.

Goguen, J. A. (1992). Sheaf semantics for concurrent interacting objects. *Mathematical Structures in Computer Science*, *2*(2), 159–191.

Goodman, A. A., & Wong, C. G. (2009). Bringing the night sky closer: Discoveries in the data deluge. *The Fourth Paradigm: Data-Intensive Scientific Discovery*, 39–44.

Greene, B. (n.d.). *String theory.* Encyclopædia Britannica, Inc. Retrieved September 09, 2018, from `https://www.britannica.com/science/string-theory`.

Gruber, T. (1992). *What is an ontology?* Retrieved December 13, 2018, from `http://www-ksl.stanford.edu/kst/what-is-an-ontology.html`.

GTAI. (n.d.). *GTAI – Industrie 4.0 - what is it?* Retrieved January 28, 2019, from `https://www.gtai.de/GTAI/Navigation/EN/Invest/Industries/Industrie-4-0/Industrie-4-0/industrie-4-0-what-is-it,t=smart-industry,did=1798434.html`.

Guarino, N. (1995). Formal ontology, conceptual analysis and knowledge representation. *International Journal of Human-Computer Studies*, *43*(5-6), 625–640.

Guarino, N. (1998). *Formal ontology in information systems: Proceedings of the first international conference (FOIS'98)* (Vol. 46). IOS Press.

Guarino, N., Oberle, D., & Staab, S. (2009). What is an ontology? In *Handbook on ontologies* (pp. 1–17). Springer.

Gunning, D., & Aha, D. (2019). Darpa's explainable artificial intelligence (XAI) program. *AI Magazine*, *40*(2), 44–58.

Hall, A. (1954). *The scientific revolution, 1500-1800: The formation of the modern scientific attitude*. Longmans, Green and Company.

Hall, M., Harborne, D., Tomsett, R., Galetic, V., Quintana-Amate, S., Nottle, A., & Preece, A. (2019). A systematic method to understand requirements for explainable AI (XAI) systems. In *Proceedings of the IJCAI workshop on explainable artificial intelligence (XAI 2019)*.

Harper, R. (2016). *Practical foundations for programming languages*. Cambridge University Press.

Harrison, T. M., & Barthel, B. (2009). Wielding new media in Web 2.0: exploring the history of engagement with the collaborative construction of media products. *New Media & Society*, *11*(1-2), 155–178.

Hatcher, W. S., & Scott, P. J. (1986). Lambda-algebras and C-monoids. *Mathematical Logic Quarterly*, *32*(25-30), 415–430.

Healy, M. J. (2010). Category theory as a mathematics for formalizing ontologies. In *Theory and applications of ontology: Computer applications* (pp. 487–510). Springer.

Healy, M. J., & Caudell, T. P. (2006). Ontologies and worlds in category theory: implications for neural systems. *Axiomathes*, *16*(1), 165–214.

Hellman, G. (2003). Does category theory provide a framework for mathematical structuralism? *Philosophia Mathematica*, *11*(2), 129–157.

Hempel, S., Pineda, R., & Smith, E. (2011). Self-reference as a principal indicator of complexity. *Procedia Computer Science*, *6*, 22–27.

Hepp, M. (n.d.). *Goodrelations: The professional web vocabulary for e-commerce.* Retrieved March 03, 2020, from `http://www.heppnetz.de/projects/goodrelations`.

Hermann, M., Pentek, T., & Otto, B. (2016). Design principles for industrie 4.0

scenarios. In *2016 49th Hawaii international conference on system sciences (HICSS)* (pp. 3928–3937).

Hey, T., Tansley, S., & Tolle, K. M. (2009). Jim Gray on eScience: a transformed scientific method. In *The fourth paradigm: data-intensive scientific discovery* (pp. xvii–xxxi). Microsoft Research Redmond, WA.

Hey, T., Tansley, S., Tolle, K. M., et al. (2009). *The fourth paradigm: data-intensive scientific discovery*. Microsoft Research Redmond, WA.

Hilton, D. J. (1990). Conversational processes and causal explanation. *Psychological Bulletin*, *107*(1), 65–81.

Hines, P. M. (2013). Coherence in Hilbert's hotel. *arXiv preprint arXiv:1304.5954*.

Hines, P. M. (2016). Coherence and strictification for self-similarity. *Journal of Homotopy and Related Structures*, *11*(4), 847–867.

Hitzler, P., Krotzsch, M., Ehrig, M., & Sure, Y. (2005). *What is ontology merging?–a category theoretical perspective using pushouts*. American Association of Artificial Intelligence Press.

Horrocks, I., Parsia, B., Patel-Schneider, P., & Hendler, J. (2005). Semantic web architecture: Stack or two towers? In *International workshop on principles and practice of semantic web reasoning* (pp. 37–41).

Hoyningen-Huene, P. (2008). Systematicity: The nature of science. *Philosophia*, *36*(2), 167–180.

Hoyningen-Huene, P. (2013). *Systematicity: The nature of science*. Oxford University Press.

Hurwitz, J., & Kirsch, D. (2018). *Machine learning for dummies - ibm limited edition*. John Wiley and Sons.

Iba, T. (2010). An autopoietic systems theory for creativity. *Procedia-social and Behavioral Sciences*, *2*(4), 6610–6625.

Irvine, A. D. (2015). Alfred north whitehead. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Winter 2015 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/win2015/entries/whitehead`.

Jansen, L. (2009). Classifications. *Applied Ontology: An Introduction*, 159–172.

Johnson, M., & Dampney, C. N. (2001). On category theory as a (meta) ontology for information systems research. In *Proceedings of the international conference on formal ontology in information systems-volume 2001* (pp. 59–69).

Johnson, M., & Rosebrugh, R. (2010). Ontology engineering, universal algebra, and category theory. In *Theory and applications of ontology: Computer applications* (pp. 565–576). Springer.

Johnson, M., Rosebrugh, R., & Wood, R. J. (2002). Entity-relationship-attribute designs and sketches. *Theory and Applications of Categories*, *10*(3).

Johnson, M., Rosebrugh, R., & Wood, R. J. (2012). Lenses, fibrations and universal translations. *Mathematical Structures in Computer Science*, *22*(1), 25–42.

Johnstone, P. T., et al. (2002). *Sketches of an elephant: A topos theory compendium* (Vol. 2). Oxford University Press.

Kagermann, H., Helbig, J., Hellinger, A., & Wahlster, W. (2013). *Recommendations for implementing the strategic initiative industrie 4.0: Securing the future of german manufacturing industry; final report of the industrie 4.0 working group.* Forschungsunion.

Kahneman, D. (2011). *Thinking, fast and slow.* Macmillan.

Kaufmann, W. J., & Smarr, L. L. (1992). *Supercomputing and the transformation of science.* WH Freeman & Co.

Keil, J. M. (2016). *IRI, URI, URL, URN and their differences – fusion.* Retrieved March 13, 2019, from `https://fusion.cs.uni-jena.de/fusion/blog/2016/11/18/iri-uri-url-urn-and-their-differences`.

Keinänen, M. (2008). Revisionary and descriptive metaphysics. *Philosophica*, *81*(1), 23–58.

Kent, R. E. (2010). The institutional approach. In *Theory and applications of ontology: Computer applications* (pp. 533–563). Springer.

Khanzode, K., & Sarode, R. (2016). Evolution of the world wide web: From web 1.0 to 6.0. *International Journal of Digital Library Services*, *6*(2), 1–11.

Kishlansky, M., Geary, P., & O'Brien, P. (2007). *Civilization in the west, combined volume.* Pearson.

Kitchin, R. (2013). Big data and human geography: Opportunities, challenges and risks. *Dialogues in human geography*, *3*(3), 262–267.

Kitchin, R. (2014). Big data, new epistemologies and paradigm shifts. *Big Data & Society*, *1*(1), 2053951714528481.

Kranz, W. (1994). *Antik felsefe: Metinler ve açıklamalar* (S. Y. Baydur, Trans.). Sosyal Yayınlar.

Kratzer, A. (2021). Situations in Natural Language Semantics. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Winter 2021 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/win2021/entries/situations-semantics`.

Krömer, R. (2007). *Tool and object: a history and philosophy of category theory* (Vol. 32). Springer Science & Business Media.

Kuiler, E. W. (2014). From big data to knowledge: an ontological approach to big data analytics. *Review of Policy Research*, *31*(4), 311–318.

Kuś, M., Skowron, B., & Wójtowicz, K. (2019). Why categories? In M. Kuś & B. E. Skowron (Eds.), *Category theory in physics, mathematics, and philosophy* (pp. 1–19). Springer.

Lambek, J. (1980). From types to sets. *Advances in Mathematics*, *36*(2), 113–164.

Lambek, J. (2004). What is the world of mathematics? *Annals of Pure and Applied Logic*, *126*(1-3), 149–158.

Lambek, J., & Scott, P. J. (1984). Aspects of higher order categorical logic. *Mathematical Applications of Category Theory*, *30*, 145–174.

Lambek, J., & Scott, P. J. (1988). *Introduction to higher-order categorical logic* (Vol. 7). Cambridge University Press.

Laney, D. (2012). The importance of'big data': A definition. *Gartner. Retrieved*, *21*, 2014–2018.

Langley, A., Smallman, C., Tsoukas, H., & Van de Ven, A. H. (2013). Process studies of change in organization and management: Unveiling temporality, activity, and flow. *Academy of management journal*, *56*(1), 1–13.

Large Synoptic Survey Telescope. (n.d.). *LSST project mission statement.* Retrieved July 11, 2019, from `https://www.lsst.org/about`.

Lawvere, F. W. (1964). An elementary theory of the category of sets. *Proceedings of the National Academy of Sciences*, *52*(6), 1506–1511.

Lawvere, F. W., & Schanuel, S. H. (2009). *Conceptual mathematics: a first introduction to categories*. Cambridge University Press.

Leonelli, S. (2014). What difference does quantity make? On the epistemology of big data in biology. *Big Data & Society*, *1*(1), 2053951714534395.

Levinson, J. (1980). The particularisation of attributes. *Australasian Journal of Philosophy*, *58*(2), 102–115.

Levinson, J. (2006). Why there are no tropes. *Philosophy*, *81*(318), 563–579.

Lewis, D. (1986). *On the plurality of worlds*. Oxford: Blackwell.

Lipton, Z. C. (2018). The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, *16*(3), 31–57.

Lowe, E. J. (1995). The metaphysics of abstract objects. *The Journal of Philosophy*, *92*(10), 509–524.

Lowe, E. J. (2016). There are (probably) no relations. *The Metaphysics of Relations*, 100–112.

Lu, R.-Q. (2005). Towards a mathematical theory of knowledge. *Journal of Computer Science and Technology*, *20*(6), 751–757.

Lu, R.-Q. (2016). A peep at knowledge science in a categorical prospect. *Frontiers of Computer Science: Selected Publications from Chinese Universities*, *10*(5), 767–768.

MacBride, F. (2020). Relations. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Winter 2020 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/win2020/entries/relations`.

Mac Lane, S. (1986). *Mathematics form and function*. New York: Springer.

Mac Lane, S. (1998). *Categories for the working mathematician* (Vol. 5). Springer Science & Business Media.

Mainzer, K. (2004). Self-organization and emergence in complex dynamical systems. *Informatik 2004, Informatik verbindet, Band 2*.

Maldonado, N., Vegas, V. G., Halevi, O., Martínez, J. I., Lee, P. S., Magdassi, S., ... others (2019). 3d printing of a thermo-and solvatochromic composite material based on a cu (ii)–thymine coordination polymer with moisture sensing capabilities. *Advanced Functional Materials*, *29*(15), 1808424.

Mandelbaum, E. (2020). Associationist Theories of Thought. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Fall 2020 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/fall2020/entries/associationist-thought`.

Manning, R. N. (1998). Functional explanation. Retrieved from `https://www.rep.routledge.com/articles/thematic/functional-explanation/v-1`.

Marcus, G. (2018). Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

Marcus, G., & Davis, E. (2020). *GPT-3, bloviator: OPENAI's language generator has no idea what it's talking about.* Retrieved January 10, 2021, from `https://www.technologyreview.com/2020/08/22/1007539/gpt3-openai-language-generator-artificial-intelligence-ai-opinion`.

Marquis, J.-P. (1995). Category theory and the foundations of mathematics: philosophical excavations. *Synthese*, *103*(3), 421–447.

Marquis, J.-P. (1997). Category theory and structuralism in mathematics: Syntactical considerations. In *Philosophy of mathematics today* (pp. 123–136). Springer.

Marquis, J.-P. (2021). Category Theory. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Fall 2021 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/fall2021/entries/category-theory`.

Marr, B. (2015). *Big data: Using smart big data, analytics and metrics to make better decisions and improve performance.* John Wiley and Sons.

Marr, B. (2016a). *Big data in practice: how 45 successful companies used big data analytics to deliver extraordinary results.* John Wiley and Sons.

Marr, B. (2016b). *What everyone must know about industry 4.0.* Retrieved June

13, 2018, from https://www.forbes.com/sites/bernardmarr/2016/06/20/what-everyone-must-know-about-industry-4-0.

Masolo, C., & Borgo, S. (2005). Qualities in formal ontology. In *Foundational aspects of ontologies (FOnt 2005) workshop at KI* (pp. 2–16).

Maurin, A.-S. (2002). Tropes. In *If tropes* (pp. 8–24). Springer.

Maurin, A.-S. (2018). Tropes. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Summer 2018 ed.). Metaphysics Research Lab, Stanford University. https://plato.stanford.edu/archives/sum2018/entries/tropes.

Max-Planck-Gesellschaft. (2012). *Bonobo genome completed.* Retrieved February 02, 2022, from https://www.mpg.de/5838534/Bonobo_genome_completed.

Mayer-Schoenberger, V., & Cukier, K. (2013). *Big data: The essential guide to work, life and learning in the age of insight.* Hachette UK.

Mazur, B. (2008). When is one thing equal to some other thing? In B. Gold & R. A. Simons (Eds.), *Proof and other dilemmas: Mathematics and philosophy* (p. 221–242). Mathematical Association of America. doi: 10.5948/UPO9781614445050.015

McDermott, D. (1993). Building large knowledge-based systems: Representation and inference in the Cyc project. *Artificial Intelligence*, *61*, 53–63.

Mealy, G. H. (1967). Another look at data. In *Proceedings of the november 14-16, 1967, Fall Joint Computer Conference* (pp. 525–534). New York, NY, USA: ACM.

Menzel, C. (2021). Possible Worlds. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Fall 2021 ed.). Metaphysics Research Lab, Stanford University. https://plato.stanford.edu/archives/fall2021/entries/possible-worlds.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, *267*, 1–38.

Milner, R. (1993). Elements of interaction: Turing award lecture. *Communications of the ACM*, *36*(1), 78–89.

Molnar, G. (2003). *Powers: A study in metaphysics.* Clarendon Press.

Moltmann, F. (2013). *Abstract objects and the semantics of natural language.* OUP Oxford.

Moltmann, F. (2019). Natural language and its ontology. *Metaphysics and Cognitive Science, Oxford UP, Oxford*, 206–232.

Moltmann, F. (2020). Natural language ontology. In *The Routledge handbook of metametaphysics* (pp. 325–338). Routledge.

Moran, D., & Cohen, J. (2012). *The Husserl dictionary.* Bloomsbury Publishing.

Mormann, T. (2010). Structural universals as structural parts: Toward a general theory of parthood and composition. *Axiomathes*, *20*(2-3), 209–227.

Nath, K., Dhar, S., & Basishtha, S. (2014). Web 1.0 to web 3.0-evolution of the web and its various challenges. In *Optimization, reliabilty, and information technology (icroit), 2014 international conference on* (pp. 86–89).

Nelson, T. (1965). *Lecture notes in Computers, creativity, and the nature of the written word.* Vassar College, New York.

Nelson, T. (1982). *Literary machines.* Eastgate Systems.

Neuman, Y., & Nave, O. (2008). A mathematical theory of sign-mediated concept formation. *Applied Mathematics and Computation*, *201*(1-2), 72–81.

Nickles, T. (2017). Scientific revolutions. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Winter 2017 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/win2017/entries/scientific-revolutions`.

Niiniluoto, I. (1999). *Critical scientific realism.* OUP Oxford.

Niiniluoto, I. (2012). On tropic realism. In *Categories of being: Essays on metaphysics and logic* (pp. 439–452). Oxford University Press.

nLab authors. (2021a). *Structure.* Retrieved June 07, 2021, from `http://ncatlab.org/nlab/show/structure`.

nLab authors. (2021b). *TopGrp.* Retrieved June 07, 2021, from `http://ncatlab.org/nlab/show/TopGrp`.

nLab authors. (2021c). *Universal construction.* Retrieved June 07, 2021, from http://ncatlab.org/nlab/show/universal%20construction.

Nöth, W. (2021). System, sign, information, and communication in cybersemiotics, systems theory, and peirce. In *Introduction to cybersemiotics: A transdisciplinary perspective* (pp. 75–95). Springer.

Noy, N. F., & McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880*.

OBO Technical WG. (2016). *The OBO foundry.* Retrieved December 28, 2018, from http://obofoundry.org.

OBO Technical WG. (2018). *Anatomical entity ontology.* Retrieved December 26, 2018, from http://www.obofoundry.org/ontology/aeo.html.

Orilia, F. (1991). Type-free property theory, exemplification and Russell's paradox. *Notre Dame Journal of Formal Logic*, *32*(3), 432–447.

Orilia, F. (2000). Property theory and the revision theory of definitions. *The Journal of Symbolic Logic*, *65*(1), 212–246.

Orilia, F., & Landini, G. (2019). Truth, predication and a family of contingent paradoxes. *Journal of Philosophical Logic*, *48*(1), 113–136.

Orilia, F., & Paolini Paoletti, M. (2017). Properties. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Winter 2017 ed.). Metaphysics Research Lab, Stanford University. https://plato.stanford.edu/archives/win2017/entries/properties.

Orilia, F., & Paolini Paoletti, M. (2020). Properties. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Winter 2020 ed.). Metaphysics Research Lab, Stanford University. https://plato.stanford.edu/archives/win2020/entries/properties.

Owens, L. (1986). Vannevar bush and the differential analyzer: the text and context of an early computer. *Technology and culture*, *27*(1), 63–95.

Panza, M. (2002). Mathematisation of the science of motion and the birth of analytical mechanics: A historiographical note. In *The application of mathematics to the sciences of nature* (pp. 253–271). Springer.

Patterson, E. (2017). Knowledge representation in bicategories of relations. *arXiv preprint arXiv:1706.00526*.

Paul, L. A. (2012). Building the world from its fundamental constituents. *Philosophical Studies*, *158*(2), 221–256.

Paul, L. A. (2017). A one category ontology. In J. A. Keller (Ed.), *Freedom, metaphysics, and method: Themes from van Inwagen* (p. 32—61). Oxford: Oxford University Press.

Pearl, J. (2019). The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM*, *62*(3), 54–60.

Pearl, J., & Mackenzie, D. (2018). *The book of why: the new science of cause and effect*. Basic Books.

Peroni, S., Shotto, D., & other Collaborators. (2018). *SPAR ontologies – BiRO*. Retrieved December 28, 2018, from `http://www.sparontologies.net/ontologies/biro`.

Peruzzi, A. (2006). The meaning of category theory for 21st century philosophy. *Axiomathes*, *16*(4), 424–459.

Phillips, R. L. (1967). Descriptive versus revisionary metaphysics and the mind–body problem. *Philosophy*, *42*(160), 105–118. doi: 10.1017/S0031819100001030

Pigliucci, M. (2009). The end of theory in science? *EMBO reports*, *10*(6), 534–534.

Poli, R. (1993). Husserl's conception of formal ontology. *History and philosophy of logic*, *14*(1), 1–14.

Poli, R. (2003). Descriptive, formal and formalized ontologies. In *Husserl's logical investigations reconsidered* (pp. 183–210). Springer.

Popper, K. (2002). *Conjectures and refutations: The growth of scientific knowledge*. Routledge.

Porter, M. E. (2011). *Competitive advantage of nations: creating and sustaining superior performance* (Vol. 2). Simon and Schuster.

Prensky, M. (2009). H. sapiens digital: From digital immigrants and digital natives to digital wisdom. *Innovate: journal of online education*, *5*(3), 1.

Quine, W. V. (1948). On what there is. *The Review of Metaphysics*, 21–38.

Radford, A., Wu, J., Amodei, D., Amodei, D., Clark, J., Brundage, M., & Sutskever, I. (2019). *Better language models and their implications.* Retrieved July 20, 2019, from `https://openai.com/blog/better-language-models`.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, *1*(8).

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Randell, B. (2013). *The origins of digital computers: selected papers.* Springer.

Rescorla, M. (2020). The Computational Theory of Mind. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Fall 2020 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/fall2020/entries/computational-mind`.

Rieder, G., & Simon, J. (2016). Datatrust: Or, the political quest for numerical evidence and the epistemologies of big data. *Big Data & Society*, *3*(1), 2053951716649398.

Rieder, G., & Simon, J. (2017). Big data: A new empiricism and its epistemic and socio-political consequences. In *Berechenbarkeit der welt?* (pp. 85–105). Springer.

Robinson, H. (2020). Dualism. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Fall 2020 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/fall2020/entries/dualism`.

Robinson, H. (2021). Substance. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Fall 2021 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/fall2021/entries/substance`.

Rodriguez-Pereyra, G. (2019). Nominalism in Metaphysics. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Summer 2019 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/sum2019/entries/nominalism-metaphysics`.

Rojek, P. (2008). Three trope theories. *Axiomathes*, *18*(3), 359–377.

Roman, S. (2017). *An introduction to the language of category theory.* Springer.

Romeijn, J.-W. (2017). Philosophy of statistics. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Spring 2017 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/spr2017/entries/statistics`.

Rose, H. (1948). The mechanical differential analyser: its principles, development, and applications. *Proceedings of the Institution of Mechanical Engineers*, *159*(1), 46–54.

Rosling, H. (2010). *Hans Rosling's 200 countries, 200 years, 4 minutes–the joy of stats–BBC Four.* Retrieved March 07, 2017, from `http://www.gapminder.org/videos/the-joy-of-stats`.

Rovelli, C. (2021). *Helgoland: Making sense of the quantum revolution* (E. Segre & S. Carnell, Trans.). Riverhead Books, New York.

Satıoğlu [Yargan], D. (2015). *A philosophical approach to upper-level ontologies* (Unpublished master's thesis). Middle East Technical University.

Satıoğlu [Yargan], D. (2017). Formel ontolojilerin dinamikliği ve esnekliği Üzerine[on dynamic and flexible formal ontologies]. In V. Kamer & Ş. Ural (Eds.), *VII. Mantık Çalıştayı Kitabı* (pp. 697–703). Mantık Derneği Yayınları,İstanbul.

Schaffer, J. (2016). The Metaphysics of Causation. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Fall 2016 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/fall2016/entries/causation-metaphysics`.

Schmidt, M., & Lipson, H. (2009). Distilling free-form natural laws from experimental data. *science*, *324*(5923), 81–85.

Schorlemmer, M., & Kalfoglou, Y. (2005). Progressive ontology alignment for meaning coordination: An information-theoretic foundation. In *Proceedings of the fourth international joint conference on autonomous agents and multiagent systems* (pp. 737–744).

Schroeder-Heister, P. (2018). Proof-Theoretic Semantics. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Spring 2018 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/spr2018/entries/proof-theoretic-semantics`.

Schwab, K. (2017). *The fourth industrial revolution.* Crown Business.

Schwitzgebel, E. (2021). Belief. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Winter 2021 ed.). Metaphysics Research Lab, Stanford University. https://plato.stanford.edu/archives/win2021/entries/belief.

Scott, P. J. (2000). Some aspects of categories in computer science. In *Handbook of algebra* (Vol. 2, pp. 3–77). Elsevier.

Seremeti, L., & Kougias, I. (2013). Networks of aligned ontologies through a category theoretic approach. *Journal of Emerging Trends in Computing and Information Sciences*, *4*(10).

Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal*, *27*(3), 379–423.

Shen, Y., Huang, P.-S., Gao, J., & Chen, W. (2017). Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining* (pp. 1047–1055).

Sheth, A., Perera, S., Wijeratne, S., & Thirunarayan, K. (2017). Knowledge will propel machine understanding of content: Extrapolating from current examples. *arXiv preprint arXiv:1707.05308*.

Shilov, V. V., & Silantiev, S. A. (2016). 'machines à comparer les idées' of semen korsakov: First step towards ai. In *Ifip international conference on the history of computing* (pp. 71–86).

Shmueli, G., et al. (2010). To explain or to predict? *Statistical Science*, *25*(3), 289–310.

Simons, P. (1982). Three essays in formal ontology. *Parts and Moments, Studies in Logic and Formal Ontology, Philosophia Verlag, Munich*, 111–260.

Simons, P. (1994). Particulars in particular clothing: Three trope theories of substance. *Philosophy and Phenomenological Research*, *54*(3), 553–575.

Simons, P. (2005). Against set theory. *Experience and Analysis*, 143–152.

Singh, D., Isah, A., & Ibrahim, A. (2012). A survey of the development of the theory of categories. *International Journal of Innovation, Management and Technology*, *3*(2), 156.

Smith, B. (2002). Ontology and information systems. Retrieved from http://ontology.buffalo.edu/ontology(PIC).pdf.

Smith, B. (2004). Beyond concepts: ontology as reality representation. In *Proceedings of the third international conference on formal ontology in information systems (FOIS 2004)* (pp. 73–84).

Smith, B. (2005). Against fantology. In J. Marek & E. M. E. Reicher (Eds.), *Experience and analysis* (pp. 153–170). Vienna: ÖBV & HPT.

Smith, B. (2014). The relevance of philosophical ontology to information and computer science. In R. Hagengruber & U. Riss (Eds.), *Philosophy, computing and information science* (pp. 75–83). Chatto & Pickering.

Smith, B., & Mulligan, K. (1983). Framework for formal ontology. *Topoi*, *2*(1), 73–85.

Smith, B., & Smith, D. W. (1995). Introduction. In B. Smith & D. W. Smith (Eds.), *The Cambridge companion to Husserl* (p. 1–44). Cambridge University Press.

Smith, D. W. (1995). Mind and body. In B. Smith & D. W. Smith (Eds.), *The Cambridge companion to Husserl* (p. 323–393). Cambridge University Press.

Smith, D. W. (2007). *Husserl*. Routledge.

Smith, J. (1984). An interpretation of Martin-Löf's type theory in a type-free theory of propositions. *The Journal of symbolic logic*, *49*(3), 730–753.

Sowa, J. F. (1999). *Knowledge representation: logical, philosophical and computational foundations*. Brooks/Cole Publishing Co.

Sowa, J. F. (2010). The role of logic and ontology in language and reasoning. In *Theory and applications of ontology: philosophical perspectives* (pp. 231–263). Springer.

Spivak, D. I. (2010). *Lecture slides of Databases are categories.* Galois Connections. Retrieved from `http://math.mit.edu/~dspivak/informatics/talks/harvard-20101103--DBCT.pdf`.

Spivak, D. I. (2014). *Category theory for the sciences.* MIT Press.

Spivak, D. I. (2015). *Categorical informatics: A functorial approach to data integration.* `https://math.mit.edu/~dspivak/informatics/grants/NSF_IIS.pdf`.

Spivak, D. I., & Kent, R. E. (2012). Ologs: a categorical framework for knowledge representation. *PloS one*, *7*(1), e24274.

Stanford Center for Biomedical Informatics Research. (2016). *Protégé*. Retrieved December 26, 2018, from `https://protege.stanford.edu`.

Steadman, I. (2013). Big data and the death of the theorist. *Wired*, *25*, 2013–01.

Steup, M. (2018). Epistemology. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Summer 2018 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/sum2018/entries/epistemology`.

Stout, G. (1947). Distributive unity as a "category", and the Kantian doctrine of categories. *The Australasian Journal of Psychology and Philosophy*, *25*(1-2), 1–33.

Stout, G. F. (1940). Things, predicates and relations. *The Australasian Journal of Psychology and Philosophy*, *18*(2), 117–130.

Strawson, P. F. (2002). *Individuals: An essay in descriptive metaphysics*. Routledge.

Symons, J., & Alvarado, R. (2016). Can we trust big data? applying philosophy of science to software. *Big Data & Society*, *3*(2), 2053951716664747.

Szabó, Z. G. (2020). Compositionality. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Fall 2020 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/fall2020/entries/compositionality`.

Taylor, D. E., & Burgess, A. (2015). What in the world is semantic indeterminacy? *Analytic Philosophy*, *56*(4), 298–317.

The Ptolemy Projects. (n.d.). *Cyber-Physical Systems - a Concept Map*. Retrieved January 28, 2019, from `https://ptolemy.berkeley.edu/projects/cps`.

Thomasson, A. (2019). Categories. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Summer 2019 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/sum2019/entries/categories`.

Townsend, J. A., Adams, S. E., Waudby, C. A., de Souza, V. K., Goodman, J. M., & Murray-Rust, P. (2004). Chemical documents: machine under-

standing and automated information extraction. *Organic & biomolecular chemistry*, *2*(22), 3294–3300.

Turing, A. M. (1937). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London mathematical society*, *2*(1), 230–265.

Turner, R., & Angius, N. (2017). The philosophy of computer science. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Spring 2017 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/spr2017/entries/computer-science`.

Van Rijmenam, M. (2013). Why the 3v's are not sufficient to describe big data. *Big data startup*. Retrieved from `http://www.bigdata-startups.com/3vs-sufficient-describe-big-data`

Varela, F. (1979). *Principles of biological autonomy*. Elsevier North Holland, Inc.

Vickers, S. (2010). Issues of logic, algebra and topology in ontology. In *Theory and applications of ontology: Computer applications* (pp. 511–531). Springer.

Vidales, C., & Brier, S. (2021). *Introduction to cybersemiotics: a transdisciplinary perspective*. Springer.

W3Schools. (n.d.). *XML attributes*. Retrieved March 03, 2021, from `https://www.w3schools.com/xml/xml_attributes.asp`.

W3Schools. (2019). *XML schema tutorial*. Retrieved March 03, 2019, from `https://www.w3schools.com/xml/schema_intro.asp`.

Wadler, P. (2015). Propositions as types. *Communications of the ACM*, *58*(12), 75–84.

Walsh, P. (2017). Categorical harmony and path induction. *The Review of Symbolic Logic*, *10*(2), 301–321.

Walton, D. (2014). *Abductive reasoning*. University of Alabama Press.

Wang, Q., & Rong, L. (2007). Typed category theory-based micro-view emergency knowledge representation. In *International conference on knowledge science, engineering and management* (pp. 568–574).

Ward, J. S., & Barker, A. (2013). Undefined by data: a survey of big data definitions. *arXiv preprint arXiv:1309.5821*.

Wegner, P. (1996). Interoperability. *ACM Computing Surveys (CSUR)*, *28*(1), 285–287.

Wegner, P. (1998). Interactive foundations of computing. *Theoretical computer science*, *192*(2), 315–351.

Wegner, P., & Goldin, D. (2003). Computation beyond turing machines. *Communications of the ACM*, *46*(4), 100–102.

Wetzel, L. (2018). Types and Tokens. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Fall 2018 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/fall2018/entries/types-tokens`.

Whitehead, A. N. (1957). *Process and reality*. Macmillan New York, NY.

Williams, D. C. (1953). On the elements of being: I. *The review of metaphysics*, *7*(1), 3–18.

Winsberg, E. (2019). Computer simulations in science. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Spring 2019 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/spr2019/entries/simulations-science`.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Wittgenstein, L. (2010). *Tractatus logico-philosophicus* (C. Ogden, Trans.). Project Gutenberg.

Wojtowicz, R. L. (2013). Sketches, views and pattern-based reasoning. In *STIDS* (pp. 117–124).

Wojtowicz, R. L. (2016). Sketch theory as a framework for knowledge management. *Innovations in Systems and Software Engineering*, *12*(1), 69–79.

Woodward, J. (2017). Scientific explanation. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Fall 2017 ed.). Metaphysics Research Lab, Stanford University. `https://plato.stanford.edu/archives/fall2017/entries/scientific-explanation`.

Yargan, D. (2020a). Is science without explanations possible? *Arkhe-Logos*(9), 109–122.

Yargan, D. (2020b). Üst düzey ontoloji inşasındaki felsefi yaklaşımlar. *Kilikya Journal of Philosophy*(1), 32–50.

Yargan, D., & Zambak, A. F. (2021). Medikal Ontolojiler. In N. Bozbuğa & S. E. Gülseçen (Eds.), *Tıp bilişimi* (pp. 26–60). Istanbul University Press. `https://iupress.istanbul.edu.tr/tr/book/tip-bilisimi/home`. doi: 10.26650/B/ET07.2021.003.02

Yates, F., et al. (1983). *What is self-organization.* Princeton University Press: Princeton, NJ, USA. Retrieved March 04, 2022, from `http://assets.press.princeton.edu/chapters/s7104.pdf`.

Zambak, A. F. (2013). The frame problem: Autonomy approach versus designer approach. In *Philosophy and theory of artificial intelligence* (pp. 307–319). Springer.

Zambak, A. F. (2014). Artificial intelligence as a new metaphysical project. In R. Hagengruber & U. V. Riss (Eds.), *Philosophy, computing and information science* (p. 67–74). Pickering & Chatto.

Zambak, A. F., & Vergauwen, R. (2007). MS models: A new approach to models and simulations in artificial intelligence. *Logique et Analyse*, 179–206.

Zeller, E. (2008). *Grek felsefesi tarihi* (A. Aydoğan, Trans.). Say.

Zimmermann, A., Krotzsch, M., Euzenat, J., & Hitzler, P. (2006). Formalizing ontology alignment and its operations with category theory. In *4th international conference on formal ontology in information systems (FOIS)* (pp. 277–288). IOS Press.

APPENDICES

A. FROM 1.0 TO 3.0: INDUSTRY, SCIENCE, WEB

The very fundamental wheels upon which the world makes progress are science and industry. Developments in each field have resulted in new stages or revolutions to shape the world's destiny. Specific to our age, the improvements in both hardware and software have changed the way we produce: Big Data helps people in science and industry thrive. The Web, another field that dominates our lives, has gained a challenging role in Big Data. The historical and philosophical background of our lives' three essential fields –industry, science, and the Web– from the machine and/or data involvement, generation and/or usage perspectives. This appendix chapter provides the details of 1.0, 2.0, and 3.0 versions of Industry, Science, and Web, upon which we built ISW 4.0s.

A.1 Industry

The term 'industry' points to economic activities that produce or provide "goods, services, or sources of income" (Encyclopædia Britannica, 2011). When there is population growth in a territory, the need for more food, more clothes, more shelters increases. The existing industries must be transformed to meet the demands. Such transformations are sometimes finding better ways to produce or harvest more or adjust the whole production style to meet the existing system's limitations. New technologies, the latest scientific discoveries, and/or unexpected social phenomena (such as the dramatic increase in population) spark industry revolutions.

We will examine Industry in distinct periods which are almost in consensus. In the history books, the first period and revolution in the industry called the *Industry Revolution*. This period is when steam engines were introduced, and immediately afterward, when mechanization of some works from which humans were liberated. Thanks to immense scientific, especially in mechanics, discoveries, and (greedy) market demand, the second period starts with electricity usage in factories. This new epoch begins with the introduction of the assembly line that called forth mass production. The third period is called the *Digital* or *Computer Revolution* that starts with the advent of computers, which led to automation in factories. Now we are at the fourth industrial revolution called *Industry 4.0* that aims at the decentralized and autonomous factories. Coined at the opening of Hanover Fair in 2011, the term "Industry 4.0" also named the previous three periods/revolutions as Industry 1.0, Industry 2.0, and Industry 3.0, respectively, making the historical progress in Industry more concrete.

Industries such as agriculture, mining, or fishing have been held since the story of humankind. In each revolutionary period, these industries have continued to exist with the change of the period provided. That is, Industry 1.0, Industry 2.0, and Industry 3.0 have surpassed the former's limitations. In this section, we will be discussing the revolutions in the history of Industry, the radical changes in the ways we live, produce, consume, and behave, as they would never be the same again.

A.1.1 Industry 1.0

By the beginning of the eighteenth century, the population of Europe was continuously increasing. The population of Britain, for example, grew by more than 80% in the early of the century (Kishlansky, Geary, & O'Brien, 2007, p. 621). Overpopulation was the cause for more food and more cloth. The traditional agricultural production, whose limits were determined by nature, could not meet these requirements. People were lack of food to consume, cloth to wear, work to earn their lives. It was an attitude change that steered agriculture, which has been of commercial concern since then. Although known and practiced for cen-

turies during overpopulation periods, various methods were used *systematically* to increase crop yields. These early systematic agricultural practices are labeled as the *agricultural revolution*, which was a revolution of technique, rather than of technology, and it shifted substance farming into commercial agriculture. The abundance was firstly provided by enclosing the lands and then by determining what and when to harvest according to the community's needs. For instance, more grains were harvested not only for humans but also for livestock so that farmers could increase their livestock and their wealth. The ability to tune the agricultural products into the market demands maximized farmers' profits, abundant raw materials, and foodstuff for urban and rural workers. Fall in food prices and an increase of free human power delivered a solution for the requirement for more cloth. Before the agricultural revolution, manufacturing was labor carried out at homes as an occasional work for supplement income, called *the cottage industry*–merchants often providing the raw materials and essential equipment, and then picking up the finished product—, that was less important and less valuable compared to farming. On the other hand, the traditional commercial cloth production, which urban artisans held, could not meet the overpopulation's cloth demand. The oversupply of labor offered the opportunity to rural families to sell their weaving skills. More and more, manufacturing became a family-oriented occupation of many rural families. Consequently, society's transition can be illustrated as: "The open-field village was a community; the enclosed estate was a business" (Kishlansky et al., 2007, p. 626).

The agriculture revolution triggered the burst of industrialization. In the first half of the nineteenth century in Europe, industrialization transformed cultures from agrarian and rural into industrial and urban (Buchanan, 2018). This transformation must be taken as a revolution rather than as development in the consequences of the agricultural revolution and of the changes in manufacturing. So, the industrial revolution,[1] Industry 1.0, in our terms, refers to "a sustained period of economic growth and change brought about by the application of

[1] The term "Industrial Revolution" is not a precise historical concept. For various instructional or conceptual purposes, historians name different periods as the Industrial Revolution. See Buchanan (2018) for some detail.

mineral energy and technological innovations to the process of manufacturing" (Kishlansky et al., 2007, p. 626). In the simplest terms, Industry 1.0 is when machines, steam engines, and the textile industry changed the Western world's dynamics.[2]

Before Industry 1.0, the major sources of power were the wind, water, and muscle, but from the nineteenth century on, powerful engines were substituted as power sources as well. For instance, as mentioned above, the cottage industry, required human power, could not meet the population's demand, then many people thought of how to improve the process of cloth production. In 1764, the spinning jenny was invented by James Hargreaves. This machine was operated on a number of spindles for spinning threads. Once the production of threads was increased, then it was time to mechanize the looms. During the cottage industry, people were using hand-looms that were for weaving threads into cloth. Invented by Edmund Cartwright in 1784 and first built in 1785, power looms were steam-powered machines that reduced the need for human power and increased the number of products.[3] Along with textile mechanization, the steam engines shifted human life into something that could never be the same again. In 1712, Thomas Newcomen developed the first steam engine used for pumping water out of mines. Many scientists of this period improved the Newcomen's technology with some adaptations and developed their steam engines used in various fields, e.g., for iron production, and then for new transportation technologies -steamships and locomotives (Kishlansky et al., 2007). Moreover, with the demands for reaching new markets and low-cost raw materials, people looked for ways to power their industrial initiatives. Many industries looked for ways to engage steam power into their manufacturing, which vivified the economy of many lands (Buchanan, 2018).

[2] The industrial revolution was emerged in the Great Brain, for she had abundant mineral resources, such as coal and iron ores, which were indispensable for industrialization; colonies that served as sources of raw material and as a market for commercial goods; and entrepreneurs, inventors, and skilled workers. Consequently, Kishlansky et al. (2007, p. 643) put "No one else had to invent the jenny, the mule, or the steam engine."

[3] The Jacquard machine, or the Jacquard loom, is a power loom that used punched cards for pattern creation. Punched cards were introduced during Industry 1.0 that is crucially important for Data 1.0 since designing the punched cards is taken as an ancestor of programming.

The story of Industry 1.0 started with overpopulation and systematic agriculture, whose indirect but inevitable effects initiated coal use and ended up with steam power usage. All these paved the way for mechanized production by powerful machines; consequently, Industry 1.0, or the Industrial Revolution, is when the power sources have changed, which animated the machines used for manufacturing, transportation, and communication.[4]

A.1.2 Industry 2.0

The emergence of mechanization in Industry 1.0 inevitably led to the factory system, which is at the heart of Industry 2.0 that is the era starting with the electrification of factories and the birth of mass production. There are two crucial energy sources for Industry 2.0: electricity and petroleum. Having widespread applications in industry, steam engines were evolved and modified with respect to requisites of a specific industry; for instance, locomotives require high-pressure steam to move such heavy vehicles (Buchanan, 2018). Although such engines were developed to meet this need, they could not provide a necessary high speed to rotate the dynamos to function effectively, so designers were looking for radical modifications or novel engine systems. Besides modifications like the Willans engine or the uniflow engine, the steam tribune designed by Charles Parson solved the mentioned problem efficiently. However, this major technological innovation prompted to generate electricity as a new power source (ibid).

Electric motors were a product of the search that how electricity could be used as a source of power in the industry. One of the electricity practices was electric traction, whose first implementation was the electric tramways and the subway systems. On the other hand, these means of transportation were not the only ones that occurred in this era: One of the most important inventions was the

[4] In the early 1800s, a group of textile workers rebelled against the introduction of machines due to the well-founded fear of losing their position as highly trained artisans and jobs. They knew that they could never compete with the speed of the devices, and above all, their craftsman was past its sell-by date. People who supported this movement called *Luddites*. Since that time, there are 'Luddites' who have been under the threat of their artisanship being substituted by machines. Today, the developments in AI increase the number of Luddites, but we cannot destroy machines in some protests as they did.

internal-combustion engine, which can be regarded as a prime mover in the nineteenth century (ibid). The significant difficulty operating such engines was finding suitable fuel, which was come over with utilizing oil. Oil has been known and used for centuries, such as for illuminants and medical products, but its intense usage for engines encounters internal-combustion engines' invention.

Meanwhile, Ford Motor Company was working on a car design that would provide inexpensive transportation to the US's middle-class citizens. Up to that time, each car was a product of craftsmen who had deep knowledge about mechanics and other car manufacturing aspects and who were highly skilled in conducting/executing their knowledge—becoming such a person requires time and effort. Then Henry Ford introduced the *assembly line* whose function was to increase productivity and decrease the cars' cost. Such a creative mind lets workers gather along with a moving line. These workers need not have deep knowledge or skills about the whole process; namely, they were not craftsmen necessarily; instead, they were unskilled and needed to perform a particular task each time. Such labor division provided quick and efficient production of goods in general, which paved the way for *mass production*. In other words, Industry 2.0 refers to new organizational models of production, which aims at the automation of the process by introducing a new industrial model of large factories run around assembly lines. To sum up, the assembly line's introduction changed the production speed, interest, and size. Industry 2.0, therefore, provided affordable goods for a large number of people by *automation the process*.

A.1.3 Industry 3.0

Industry 1.0 bestowed the invention of the steam engine and mechanized production; by the advent of electricity and the invention of the assembly line, Industry 2.0 provided mass production. It is crystal clear from the facts of these periods that the scientific discoveries and the new technologies called forth industrial developments, the need for human-power radically diminished, the human control during production was gradually evanesced, and finally, more and new products were produced. Then, in Industry 3.0, we would expect the same pattern: scientific studies would lead to new technology, automation would be expanded,

and mechanical processes would rule the progress, and new products would be on the markets thanks to all these improvements.

The third industrial revolution, namely Industry 3.0, starts with the advent of computers, thereby it is also called *Digital Revolution* or *Computer Revolution*, and this period is typically called the *Information Age*. What paved the way for Industry 3.0 along with scientific studies on computation that have been held for centuries are semiconductors -transistors-, mainframe computers, home computers, and the Internet (Schwab, 2017). Brief information about the history of computers is essential for a better understanding, so below it is.

The story of the ambition of mechanizing reasoning and computing, which dates back to Leibniz, if not earlier. He aimed at creating *calculus ratiocinator*, a machine that computes the validity of arguments. Almost a century later comes Babbage started to design the *Analytical Engine*, a machine for mechanizing reasoning, at least as Ada Lovelace thought of (Boden, 2016, pp. 7–8). The Analytical Engine, or the mechanical general-purpose computer, was the first idea of a programmable machine. It was never completed at the time of Babbage due to the fact that electronic components, especially microelectronic devices, were required for its construction (Freeman & Louça, 2001). About seventy years later of the Analytical Engine's first introduction, the first electro-mechanical machines were launched by Zuse (Copeland, 2017). In 1939, Atanasoff and Berry introduced the first electronic computer; in the 1940s, the University of Pennsylvania built ENIAC, EDVAC, and UNIVAC that led the computer industry in the States (Freeman & Louça, 2001). Meantime, IBM was dominating the market by providing tabulators and punched cards[5] to departments in the US government and the private sector. It was when von Neumann developed his computer architecture, and the computer industry accelerated rapidly. In the 1960s, IBM sold the mainframe computers to various countries; in the 1970s, minicomputers were started to be realized by replacing mainframe computers, and since the 1980s, they became a part of our daily lives. Meanwhile, in 1969

[5] See section Data 1.0.

the Internet was introduced to some universities for file exchanging, then to the world for data exchange. Moreover, the nascent electronic industry boosted the industries of mobile phones, software, and microelectronics.

The ambition of mechanizing reasoning, although it has not been fully accomplished, and the advent of transistors gave birth to computers. Led by transistor technology, computers have been programmed by machine languages that enable us to tell a machine what to do. That is, there have been machines that are programmed for specific tasks. The advent of computers is by itself an industrial development since it comes with many industries, such as microprocessors, software developments, or the mobile phone industry. Computers became indispensable tools for almost all jobs; accordingly, the software industry rocketed as well, for there has been a need for various programs that can operate for specific works- those works used to take lots of time before using computers. Foremost, what makes Industry 3.0 a revolution over Industry 2.0 is the advent of *automation*: The robots have replaced workers on the assembly lines,[6] those are, in general, machines, which are programmed in order to carry out a series of actions automatically. Then, machines in Industry 3.0 are fine-tuned versions of the previous periods: They can be externally controlled or perform their specific tasks uninterruptedly. Once again, the revolutionary incident of the era of digitization is the automation of factories by industrial robots. Today, in many factories, there are *masterly* designed robots performing their task *craftsmanly*. When the property of performing a task craftsmanly can be attributed to machines, why not could they run a factory by becoming masters of the factory? Once the workflow and other detailed plans of running a factory are put forth, an (or some) adept machine(s) can decide on what actions should be taken to run the factory. This dream, or to some people, what is happening right now, is about having autonomous factories. Automation via robots would become only a minor aspect of such factories since the automation is held by some other machines, by the autonomous machines. Realization of this and beyond can

[6] Here, we are speaking of industrial robots, and there are also software robots, patient assist robots, and nanorobots to name some.

be another revolution in Industry, which we discuss in chapter Autonomy and Data.

A.2 Science

As one of the highest human enterprises that shape the world, science is a complex system with its own methods for producing new knowledge. Its very characteristics are systematic observation, experimentation, reasoning, construction of hypotheses and theories, and testing them. Data collection can be held in a natural setting or laboratory settings or from simulations; reasoning methods can vary in different scientific works; the way of testing hypotheses can vary; yet the very aim of science, which is knowledge production, stays the same. In light of this fact, Andersen and Hepburn (2016) mention that the methodology of science aims to find the methods that pave the way for scientific knowledge production. Indeed, these methods have changed the course of scientific endeavors across history. In order to elaborate on different scientific eras, out of many scholars who have categorized the history of science into different phases, to guide us, we chose formulations of Hoyningen-Huene (2013) for his encapsulating views on science; and of Gray (2009) for that his ramification of the progress of science from a computational perspective. In this section, we will be speaking of the transformations of science characteristics towards our point of view.

Hoyningen-Huene starts his *systematic* investigation of the history of science with the question of what science is (Hoyningen-Huene (2013), see Introduction).[7] To answer this question, he highlights two aspects: The epistemic ideal and the source of scientific knowledge.

The first phase stretches from Plato and Aristotle to the seventeenth century. The epistemic ideal was "the absolute certainty of knowledge," in other words, *episteme*. *Episteme* is gained by deductive proofs based upon first principles,

[7] His systematic preliminaries are (i) *science* is taken in the broadest sense; (ii) the main focus is scientific knowledge, rather than the influence of science in other domains of life; and (iii) the criterion, the question of what science is, is used as a demarcation tool that separates science from metaphysics and pseudo-science.

which are evident axioms. Then, science is the accumulation of those absolute certainties. The second phase starts in the seventeenth century and ends in the second half of the nineteenth century. The certainty of scientific knowledge was the epistemic ideal for this phase as well, yet another reasoning method, induction, was introduced. A set of rules was defined as the scientific method to secure knowledge, which guarantees that laws can be achieved from data by applying those procedures correctly. This phase ends with the suspicions upon the scientific knowledge's certainty, although such knowledge is auspices of scientific methods. Then comes the third phase till the late twentieth century. In this phase, the epistemic ideal, the certainty of scientific knowledge, was toppled, and its place was taken by fallibility. The discovery of non-Euclidean geometries discredited the absolute truth of mathematics. So, mathematical claims were considered only as conditional truths of theorems. The process of erosion of scientific certainty supervened upon this. The emergence of relativity theory and quantum mechanics forced the scientist to think of novel ways to explain the phenomena rather than explaining them in terms of classical mechanics. As a consequence, scientific methods no more include strict procedures for confirmation or testing of hypotheses. From the last decades of the twentieth century, we have been in the fourth phase. At this ultimate phase, both the certainty of scientific knowledge and the existence of the scientific methods as strict rules have already eroded. In this respect, some specific research problems in a specific context are sometimes conveyed under scientific methods that cannot be taken as universally valid methodological rules. Thereby, there cannot be a consensus on the scientific methods, according to science's nature.

Jim Gray divides scientific studies into four phases concerning the evolutional aspect of science (Hey, Tansley, & Tolle, 2009, p. xvii-xix). According to this criterion, science had to be primarily empirical. In the first phase, the scientific activities were conveyed by observations and experimentations, then rationalized the findings, and finally, descriptively reported them. The second phase was theoretical, meaning that scientists of this era were making generalizations about phenomena they were examining and then constructing models of these phenomena and expressing their conclusions in a mathematical language. Since

these models and generalizations became too challenging to be solved analytically, computers have been used across the scientific spectrum and thus have become the third leg of science. That is, science has become computational, which has acted as a defining characteristic of the third era. Today, in its fourth phase, science is data explorational, for scientists "are 'looking' through large-scale, complex instruments which relay data to data centres, and only then do they look at the information on their computers" (Hey, Tansley, & Tolle, 2009, p. xix). For instance, third-phase biologists who conducted experiments in "wet labs" were experts in using glassware on benches along with using machines for doing certain calculations. In contrast, the fourth-phase biologists are considered experts only when they are skillful in working with algorithms to analyze complex data that is no longer collected from wet labs. That is to say, the widespread use of Big Data and the prowess of new technologies in data analyses have shifted science into its fourth phase.

To make a concrete example, consider studies in astronomy. At the empirical astronomy period, scientists were looking at the sky either with naked-eye or through telescopes and expressed their findings in a declarative way; at the theoretical astronomy period, their observation techniques did not change, but the way they expressed their findings did. Consider Kepler, who formalized the planetary motion in a mathematical formula. At the computational astronomy, scientists no longer look through telescopes with the naked-eye but collect data from telescopes with charge-coupled devices, detectors at non-optical wavelengths, and from running simulations. Thus, astronomy converted from 'an observational science' to 'a digital and computational science' (Hey, Tansley, Tolle, et al., 2009, p. 40). Nevertheless, today, the images collected by such devices are used to build models and simulations, which help astronomers test their hypotheses and/or change them. That is to say; astronomers are to use the machine for data generation, data processing, experiments, testing the hypotheses, and all the other scientific methods that a scientist in astronomy must follow. Never do they look at the sky; but the screens, all their scientific works are held in the machine.

Examining these two different ramifications, we can highlight two common critical transformations: the development of scientific instruments and the expansion of data sources. These transformations compose our criteria for scientific phases. In light of this, in the first phase, the scientific observations were done without an instrument; the experiments were in vivo; the scientific results were generated by reasoning alone. The second phase began with the inventions of measurement tools and the experimental instruments that enabled unobservable becoming visible and the experiments being holding both in vivo and in vitro. Such technologies also increased the volume of collected data. The third phase started with the advent of computers. Apart from their being measurement tools, computers started to process data on behalf of us, and also, they have contributed to data production. That is, the scientific experiments have been enlarged to in silico by computer simulations.[8] For the fourth phase, one can say that Big Data technologies have enormously increased the amount of data, and data analytics contribute to scientific knowledge production. In the following, we elaborate on the first three phases, respectively. We examine whether there is a fourth paradigm in science in chapter Autonomy and Data.

A.2.1 Science 1.0

The first phase of Science, Science 1.0, is the primitive way of generating scientific knowledge. The data is obtained through careful passive observations of phenomena, which takes place *in vivo*. Observation, indeed, is dealing with details of the resulting *perceptual experience* (Bogen, 2017). The observed data is processed by reason, the results are validated through deductive logic laws, and scientific descriptions are expressed in narrative features (Cf. Andersen & Hepburn, 2016).

The scientific enterprise of Aristotle is a good example of scientific knowledge production of this phase. Aristotelian inquiry starts with what is knowable

[8] There are no sharp distinctions between the periods. Today, we can still have scientific knowledge produced from the second period's methods, yet it cannot reside the information beyond what a high school student produces.

by us, which is sensory data; thus, any human investigation must begin with an observation. This starting point is passive, viz., the experiments are not controlled (Andersen & Hepburn, 2016, p. 7). The Aristotelian inquiry aims at arriving at what is knowable in nature, i.e., acquiring the genuine *episteme* about the issue at hand. Finally, the observed data are reasoned in light of Aristotelian doctrines. Panza (2002, p. 257) clarifies Aristotle's methodology as follows.

> [...]for Aristotle motion is a natural primitive phenomenon that can be characterized in general in term of the fundamental metaphysical categories of fulfillment and potentiality and described in its particular manifestation by specifying the particular body which is moving, the causes of its motion, the path this body covers, the time it takes to do it, and the speed with which it does it.

In order to construct a scientific body, the observations and the results of experiments must be appropriately arranged. The explanations of these hinge on, first of all, the principles that the scientist committed to. Lastly, the scientific body is expressed by nomenclatures, and/or by taxonomies, and/or by classifications. In the example of Aristotle, the motion is explained through doctrines of actuality and potentiality and four causes. Under the light of these metaphysical principles, the observable features of particular bodies' motion are categorized by applying the reasoning method, the truth-transferring deduction theory.

In sum, the knowledge generation in Science 1.0 is through passive-yet-careful observations in nature, or say experiments *in vivo*, and then seeking rules or applying first principles to explain the phenomenon at issue (Andersen & Hepburn, 2016).

A.2.2 Science 2.0

Science 1.0 has a limited source of data, the data obtained from the passive observations. Moreover, such perceptual experiences are object to limitations of the senses. While Science 2.0 involves the inventions of measurement tools, instruments, and experimental instruments that have enabled us to collect more data about and from the phenomena at hand. For instance, as a scientific instrument, a telescope helps us perceive distant objects; as a measurement tool,

a sextant measures the angular distance between two objects; as an experimental instrument, a graduated cylinder measures the liquid volume. These scientific instruments and measurement tools are used to collect data *in vivo*, whereas experimental instruments are used for collecting data *in vitro*; namely, scientists conduct experiments in the laboratory settings in a controlled manner. This very situation widens the scientific knowledge, which is the most remarkable difference between Science 1.0 and Science 2.0.

In vivo, it is quite challenging to do experiments since the external conditions directly affect the outcomes. However, in vitro, as its name suggests, experiments take place in isolation, where the observed objects are organized and manipulated for scientific knowledge production. For instance, observers need not wait till spring comes. They can arrange laboratory settings to the desirable conditions. Further, some phenomena in nature, such as neutrinos, cannot register on either the senses or the experimental equipment; then, observers could not produce any data. On the other hand, conducting an experiment in which neutrinos interact with chloride produces radioactive argon isotope, whose radiation can be measured by Geiger counter. That is how scientists make unobservable in vivo observable in vitro: produce data and calculate its properties indirectly. This is revolutionary!

The benefits of scientific devices' inventions cannot boil down to the vast increase in data volume. The scientific instruments collect data in a structured way, and the data are ready for being analyzed in a mathematical fashion. The structured feature of the data, both observed and gained, are used for finding patterns in it. For instance, Tycho Brache's records were used by Kepler to formulate his laws of planetary movement. That is to say, in this phase, the language of science became mathematics, and ever since, scientific communication has been conducted through formalism. This also extends the ways of processing data.

Moreover, inductive reasoning has become prevalent in this phase since we gain the truth of a phenomenon through observations, or more precisely, through structured data. Therefore, not only the amount of data or the ways of pro-

cessing data has increased, but also various reasoning methods have become dominant in scientific knowledge production.

A.2.3 Science 3.0

The advent of computers is one of the most significant cornerstones of humankind.[9] From the first day of their occurrence, the ways we play games, trade, communicate, or learn have dramatically changed. It must be crystal-clear that computers have become a scientific device in scientists' toolbox, and the integration of computers into scientific endeavors should have transformed the data sources. Our examination of computers' contributions to Science is fourfold: computers as calculators, as observers, as scientific experimental settings, and as data processors. These contributions bring forth Science 3.0.

In Science 2.0, the measurement tools have provided scientific data to be stored structurally and formalism helped these data be processed easily and efficiently by human efforts. The formalization of scientific work resulted in traditional ways of computing not being able to draw solutions. Many researchers of the day were trying to invent machines that could solve complex mathematical problems that either took so much time or could not be solved using formal methods. For instance, many scientific and engineering problems represented in mathematical models could only be solved using differential equations. These notorious equations were too tough to be solved by human computers, and at many times solutions by hand were labor-intensive processes and error-prone; these results were vital in practical scientific applications. To provide numerical solutions to those mathematical models, Bush and his coworkers at MIT developed, for instance, the Rockefeller Differential Analyzer, "a great electromechanical brain ready to tackle the problems of peace and to advance science by freeing it from the pick-and-shovel work of mathematics" (Owens, 1986, p. 64). There were many other mechanical and electromechanical differential analyzers, which were applied in scientific solutions. For instance, Morse and Allis used it to study the

[9] Both in Industry and Science, the interchangeable use of the words "computer" and "machines" may be misunderstood. To prevent this, we mention machines as computers time-to-time in this part.

scattering of electrons by helium atoms; Jackson and Tyson used it to calculate energy exchange between a gas and a solid at a different temperature (Rose, 1948, p. 52). The advent of computers fostered the usage of machines in scientific calculations. Without such a device, humankind could not have landed on the Moon.

The scientists of Science 1.0, as passive observers, note down their observations; whereas, the scientists of Science 2.0, as active observers, note and record their observations. In Science 3.0, computers have been gathering data as if they were the observers themselves. For instance, astronomy researchers of the second phase used telescopes, and then analog photographic plates to record their observations of the sky. These data were recorded and processed by scientists. In the second phase, during the late days of astronomy research, photographic plates, augmented with electronic devices, amplified the signal from celestial objects. This revolution in astronomy came with the invention of the charge-coupled device (CCD) that detects even non-optical wavelengths and then records their discoveries in digital databanks. That is to say, more data became available to astronomers, which "transformed astronomy from an observational science into a digital and computational science." (Goodman & Wong, 2009, p. 40). Results gathered from recent research can be given as a second example. Maldonado et al. (2019) developed a sensor-material for environmental monitoring, which detects the amount of water in organic solvents. Machines sending X-rays examine the sensor-material to record humidity levels. The data generated by this system will also be used in other experiments. Therefore, it can be concluded that machines have been making considerable contributions to data generation; this can also be considered as a revolution in scientific devices.

Other means of data generation in the third phase of science are computer simulations. According to Winsberg (2019), a computer simulation is a program that is designed to explore the behavior of a natural phenomenon. This mathematical model mimics a target system, which contains analytically unsolvable equations. On the other hand, this is not the only reason for creating numerical representations of the natural world.

Computer simulations are modeled for three purposes (Winsberg, 2019). The first one is for predicting how a system in the real world can behave in particular situations. Scientists can observe a model of interest in particular settings for predicting the future or retrodicting the past. Numerical weather prediction is a good example of this kind of simulation since it is built upon mathematical models. The second purpose is that computer simulations can be used to communicate with other researchers and to represent the models for a better understanding of themselves. The ultimate purpose of creating computer simulations is to understand systems and their behavior. These kinds of simulations are used for comprehending the events in systems and are constructed in order to understand the potential of specific events of occurring or how they truly happen.

One of the earliest computer simulations used in science dates back to the invention of the first general-purpose computers. Built-in 1946, ENIAC's first use was to run the simulations that helped develop the hydrogen bomb (Kaufmann & Smarr, 1992, p. 30). Although it is a notorious attempt for some scholars, a recent example of computer simulations in science is the human brain simulations. Began in 2013 and founded by European Union, Human Brain Project aims at building a human brain in silico. Scientists can do experiments with multi-level models of the brain to study various kinds of scientific questions in their minds (*Brain Simulation Platform*, 2017).

Computer simulations, then, help scientists understand complex phenomena, do experiments that cannot be tested, or cannot be conducted either *in vivo* or *in vitro*. Their advantage is rooted in the fact that a vast number of calculations or step-by-step methods can be carried out without any toil of direct observation or active recording by researchers. Thus, exploring nature with a computer has expanded scientific endeavors considerably. Therefore, the machine has excelled science into a new level by making calculations and conducting experiments *in silico*.

Last but not least, these two contributions –complex problem solvers and data generators–, with the aid of technological developments, shifted science into a

further stage, in which the machine has become data processors. Let us illustrate the impacts of computerization on science with examples. The first one is about a macro scale: observations of the biggest entities. In the second phase of science, John Campbell invented the sextant to measure latitude for navigational purposes in 1757. During the third phase of science, this tool was converted into satellite navigation, a tool that receives signals from orbiting satellites to detect the position, speed, and local time of the user. Then, conducting science of navigation or conducting science that includes navigational knowledge does not change; yet, the features of data collection and experiments have changed radically. Large Synoptic Survey Telescope (LSST) is producing a decade survey of the sky to understand and explore the structure and evolution of the universe and what it contains (Large Synoptic Survey Telescope, n.d.). The data it collects per night is about 20 terabytes, so by the end of ten years of operation, there will be about 60 petabytes of data to be processed. These vast datasets covering the whole sky are being processed by the machine that executes "automated data quality assessment and automated discovery of moving or transient sources" (ibid). This machine, both the telescope itself and the software that analyzes the dataset, goes far and far away from the human capability of data collection and reasoning.

The second example is about a micro scale: observations of the smallest entities. In the second phase of science, the devices that magnify objects were designed. Galileo Galilei is credited as the inventor of the first microscope. Human curiosity about whether there are parts smaller than what microscopes have shown led scientists to theoretical and practical searches. To make a long story short, scientists found that there are subatomic particles that do not have mass on their own, but they gain mass when they are interacting with the Higgs field (CERN, 2019). Such a field can be shown by the presence of an associated particle, which is The Higgs boson for the Higgs field because all fundamental fields have their associated particle. The experiments with ATLAS and CMS detectors at the Large Hadron Collider showed that there is a new particle that meets the theoretical expectations that Higgs would be found. Thus, showing such an existence is a "visible manifestation of the Higgs field" (ibid.). This story, on the other

hand, is a story of without-the machine-we-cannot-conduct-the-experiment-and-analyze-the-data. The experiments required colossal machines: particles cycled through the colliders about 48,000 times per second; particles collided inside the detectors more than 2 million times per second; and a trillion collisions produced only one Higgs boson (*Fermilab and the Higgs Boson*, 2019). Furthermore, in order to claim the discovery of the Higgs boson, which is nothing but what the data indicate, the dataset consisting of 800,000 simulated events must be analyzed (Adam-Bourdarios et al., 2015). This may seem like any other dataset, but the difficulty is not the dimension but the complexity of the data. There was even an open competition for analyzing this dataset: Higgs Boson Machine Learning Challenge. Without machine learning algorithms, the discovery of the Higgs Boson was impossible (Adam-Bourdarios et al., 2015).

In the chapter called *In the Emergence of Digital Science*, a quarter of a century ago, Kaufmann and Smarr (1992, p. 4) support the idea that involvement of computers -especially computer simulations- into scientific endeavor by claiming that

> No scientist could digest the vast columns of numbers that stream from the computer programs used in simulation [p. 3]. [W]ith the help of supercomputers, we will see not only these machines have taken us so far, but where they can take us in the future [p. 23]. [T]he development of digital computers has transformed the pursuit of science because it has given rise to a third methodology: the computational mode. The intend of this mode is to solve numerically the theorist's mathematical models in their full complexity. A simulation that accurately mimics a complex phenomenon contains a wealth information about that phenomenon.

Without the software, we could never collect the data, and we would never be able to analyze it or come to conclusions. Scientists have to rely on the results that the machine generates; therefore, these results are considered to be scientific knowledge. Thus, what has been revolutionary in science is that the machine is involved in both data generation and data analyzing processes; in other words, the machine has become indispensable in scientific knowledge production.

The advent of computers and their involvement in scientific knowledge production revolutionized science. In view of this, the machine has become indispensable tool that has expanded the scientific cycle in many aspects. Nevertheless, today science faces some troubles. Firstly, data analytic methods are analyzed

using huge volumes of data, which are statistical and probabilistic models in their nature. Data analytics cannot answer the why-questions, and answering those is the very essence of doing science. Secondly, among these data deluge in science, the machine could list the complex and hard-to-discover relationships, which are not bare to the scientists. On the one hand, this can be deployed into data analytics when semantic relations are introduced to the specific datasets; on the other hand, building ontologies to specify those relations is not practical under the data deluge. Thirdly, the interdisciplinary collaborations and their new types of near-real-time publishing cause scientists to hardly follow the studies done in their field. All these issues and beyond are addressed in chapter Autonomy and Data.

A.3 Web

The scientists in CERN had had trouble sharing information among them, keeping track of massive projects, reaching technical details of ex-projects, or finding a recorded information stored in the computers (Berners-Lee, 1989). To solve such issues, the vision Tim Berners-Lee had was that "all the information stored on computers everywhere were linked," and every computer could be programmed to provide a space in which everything could be linked, so that "all the bits of information in every computer at CERN, and on the planet, would be available to" everyone (Berners-Lee & Fischetti, 2001, p. 4). To realize this aim, in 1989, Berners-Lee proposed 'Information Management: A Proposal'; the studies on that single, global information space that we know today as the World Wide Web, or the Web, were just initiated.

This invention, however, cannot be attributed to Berners-Lee alone. There had been many visionaries and scientists who paved the way for this invention. Vannevar Bush ([1945] 1979) in his article "As We May Think" speaks of a hypothetical machine called *Memex*, which could create and navigate cross-references between documents in a microfilm library. Influenced by Bush's vision like many people in those days, Ted Nelson started to work on a computer-based version of *Memex*. He invented hypertext, which means "a body of written or pictorial

material interconnected in such a complex way that it could not conveniently be presented or represented on paper" (Nelson, 1965, p. 96). *Memex* can be thought of as a proto-hypertext machine. Nelson (1965) proposed another hypothetical machine called "Literary Machines" which could write and publish in hypertext. The importance of the idea is that all the information in the world could be connected so that a reader could navigate among a text or texts just by following the links. Douglas Engelbart, then, wanted to use the hypertext technology as a community collaboration method (Berners-Lee & Fischetti, 2001). For this purpose, in the 1960s, he created a collaborative workspace called oN-Line System (NLS), which was the hypertext's first practical use. "[T]he next great development in the quest for global connectivity was the Internet, a general communications infrastructure that links computers together," whose pioneers were Donald Davis, Paul Barran, Vint Cerf, Bob Kahn (Berners-Lee & Fischetti, 2001, p. 6). Such successive and accumulative ideas and technologies allowed Berners-Lee "to marry" hypertext and the Internet together to birth the Web. Consequently, the Web was ready to provide solutions to the troubles mentioned earlier, foremost for automatic information-sharing not only between scientists at CERN but also around the world.

The story of the Web starts with a dream and continues with passion. To concrete this claim, we will introduce the gradual developments in Web technologies.[10] Web 1.0 is the 'space' where people can "write and share their ideas;" Web 2.0 is where people-to-people communication and dynamic data sharing occur; Web 3.0 is where the machine can analyze the data on the Web (Cf. Berners-Lee & Fischetti, 2001).

[10] Web X.0s are defined by World Wide Web Consortium (W3C) , and we follow their definitions and explanations unless otherwise is introduced. Besides, Web 3.0 is defined as decentralized web or there are Web 5.0 and Web 6.0 in the literature; see Khanzode and Sarode (2016), for instance. On the contrary, we believe that Web 4.0 is the latest technology that has not yet been introduced so far.

A.3.1 Web 1.0

Web 1.0 starts with the birth of the Web, of no wonder, the aim of which was to make data accessible to any computer by networking.[11] Two existing technologies were sufficed to reach this aim: hypertext technology and the Internet. The hypertext technology provides the body of data, which can reside in the same or a remote computer, is connected by links so that one can reach a resource through another one.[12] On the other hand, the Internet provides access to some data that locate in a remote computer. This is the basic picture of Web 1.0; now it is time to explain how it works.

A content producer can create a web 'page,' namely a location where the content is stored. For it, she uses the language, the HyperText Markup Language (HTML), to represent the webpage's contents. Indeed, HTML provides the format of the pages containing hypertext links. Once the page with contents (text, audio, or graphics, for instance) connected by hyperlinks is created in HTML, a user can reach the page by giving the URI to a web browser, a program that shows contents in the Web in a human-readable format. This is how the content readers reach a created webpage. Now, let us explain the background processing.

Any abstract or physical resource in the Web is identified by a URI, the acronym of Uniform Resource Identifier, which can be thought of as the unique Internet address of that resource.[13] A body of data can be gained from a remote computer using its URI through a web server. A web server is software that serves the contents to the Web and allows any request to reach that content under the

[11] One caveat is necessary here. In Berners-Lee or Nelson's original texts, the concept 'information' is used for mentioning the documents stored in the computers. However, this dissertation is very sensitive to the conceptual difference of 'data' and 'information', it is correct to use 'data' here. For more information, see section Definitions of Data.

[12] The notion of 'resource' in the Web literature refers to anything; it can be a webpage, a word, a phrase, or an image. We prefer to keep this usage throughout this section. On the other hand, from a broader view, we are prone to denote those either as data -from a computational view- or as entities -from an ontological view.

[13] This caveat is for the ones who have limited knowledge about this technology. URIs are often confused with the addresses of webpages, yet they are not limited to the Web, however. A URI identifies a resource on the Internet by using a set of syntax rules to sustain uniformity all over the Internet, which obviously includes web technologies. For more information, see Berners-Lee, Fielding, and Masinter (2005).

Hypertext Transfer Protocol, HTTP. This protocol allows retrieval of linked data across the Web and fulfills the requirements of a hypertext system. A web browser reaches the given URI content and displays it in the human-readable format specified with HTML.

Web 1.0 is often called 'informational web' or 'information portal' since users can only search, read, and share contents over webpages. The webpages of those days were created by only the professionals and were stand-alone applications. The static nature of such webpages, thus, cannot allow any interaction between the generated content and the consumers. This fact shows that Web 1.0 ignores the power of the network effect, which means there are fewer content creators but more consumers. Thereby any contribution that the consumers could form was neglected (Nath, Dhar, & Basishtha, 2014). This runs into a contradiction with the goal of the Web, since the Internet and hypertext technology by themselves did not suffice user-interaction. For this reason, new technologies should be introduced for upgrading Web 1.0 from only-content-delivery nature to more-human-interaction- nature.

A.3.2 Web 2.0

Web 2.0, also called the 'Social Web', indicates that any web-user can be a content generator, and can interact with other users, thereby socialize in the 'virtual world' through the Web (Cormode & Krishnamurthy, 2008). As mentioned earlier of Web 1.0 (static webpages and passive consumers), the limitations were surpassed by introducing dynamic and interactive web experiences. This is, indeed, of the aims of the Web: "The Web is more a social interaction than a technical one" (Berners-Lee & Fischetti, 2001, p. 123).

Since the late 2003-early 2004, the webpages of Web 2.0 started to emerge (Cormode & Krishnamurthy, 2008). The shift from one-way communication of the client-server model to read-write interactive and dynamic webpages was due to attitude changes in business and user demands towards the Web and to the technological developments. Firstly, the Web became a platform where everything and everybody could meet, and such a platform has contained Web-based

communities, blogs, social networking, podcasts, RSS feeds, vlogs, and alike. With this, the users of Web 2.0 applications have not been only consumers but also participants. Secondly, various new technologies have emerged for maximizing the *content creation* and user participation. Indeed, this new platform required software artifacts to enable interactive and dynamic systems, thereby many software firms developed such technologies.[14] For instance, medium data richness was gained through XML, eXtensible Markup Language, by user labeled content; so that light interlinked data started to occur in the Web.[15] Moreover, many innovative websites, such as Facebook, YouTube, LinkedIn, have been built on those technologies to provide dynamic and interactive environments to the web-users.[16] Thirdly, Web 2.0 became user-centric. Content management, cumulative content generation, and content exchange have come into prominence since the Social Web (Harrison & Barthel, 2009). Besides, web browsers kept pace with Web 2.0 by advancing syntax-aware browsing and searching capacities.

Some of the limitations of Web 2.0 can be listed as hacking threads, ethical issues concerning the contents of the pages, limited knowledge sharing and interaction, the inability of reaching clean definitions of resources of dynamic webpages, and whether the resource them has changed and the inability of classifying the existing data on the Web (Chan, Lee, & Lin, 2009; Cormode & Krishnamurthy, 2008). Within this dissertation's frame, the most significant limitation of the 'read-and-write-web' was the data classification. Web 2.0 was developed to increase human collaboration in the Web so that web users could share their contents of any kind. The Web has become a vast databank that was an open base to be analyzed for various purposes. Besides, among the vast linked data, it was quite impossible to retrieve the data desired. Consider that one would

[14] Mashups are the popular innovations that enable combining or rendering the content in a novel form; namely, they provide cross-site links between semi-structured databases. Another essential technology used in Web 2.0 is AJAX (autonomous Javascript and XML), which is "a mixture of several technologies that integrate Web page presentation, interactive data exchange between client and server, client-side scripts, and asynchronous update of server response" (Cormode & Krishnamurthy, 2008).

[15] XML will be explained in detail in From 1.0 to 3.0: Industry, Science, Web

[16] In addition to this, unlike Web 1.0, the content readers have become the customers, and they became "first-class objects" (Cormode & Krishnamurthy, 2008). That is to say, not only is the content that people share on the Web necessary, but the very themselves are crucial for the social networks

like to find content about jaguars. In Web 2.0, they had to dig into the Web in order to find what they was really looking for: jaguar the animal, Jaguar the car, Jaguar the software, or Jaguar, their favorite band. The search results were not arranged according to their appetite, instead of descending order of a number of hits. To wit, in Web 2.0, the machine could process, convert and/or transfer the data, yet they could not 'read' the content. Hence, to upgrade the machine from being data processing and transferring tools and reach more organized 'information' on the Web, there would be an upgraded version of Web 2.0.

A.3.3 Web 3.0

Web 3.0 or *Web of Data* or the *Semantic Web* was emerged due to the need to organizing data 'meaningfully' in the *Web*, thereby the machine can 'read' the content.[17] The Semantic Web, in a nutshell, is a web of data that can be processable, or in other words, be read by the machine (Berners-Lee & Fischetti, 2001).[18] Processable data means, to our understanding, that any representation of a phenomenon is digitized in a particular way so that the machine can process it.[19] On the other hand, we are discussing here is *a web of connections* between different forms of data. This is obviously more than processing a representation: Web 3.0 processes the data interrelatedly with their meanings (Berners-Lee & Fischetti, 2001).[20]

Hypertext Markup Language, HTML, is the standard markup language for contents designed to be displayed in a web browser. Its universal codes only specify the *format* of the pages, such as the size of the header, the location of the images, or the linked resources. HTML sufficed for Web 1.0 and Web 2.0, for it is all about displaying the pages, and by itself cannot provide personalized web and powerfully expressed data. If the Web aimed to reach cumulative knowledge,

[17] See section Data 3.0.

[18] We disagree that Tim Berners-Lee (2001) names Semantic Web as the machine-understandable web. For details see sectionData 4.0.

[19] See section Data 1.0.

[20] See sectionData 3.0.

then it was to overcome the obstacle of powerful expressiveness of the pages that had been in the content. In addition to this, search engines before Web 3.0 could not link the related data, instead showed the search results by the number of clicks. Thus, there must be ways to reach the content of the pages somehow. To this end, there should be an infrastructure that could interchange and integrate the data on the Web, by which data could be exchanged and merged on a Web-scale.[21]

The task of involving semantic aspects of the data in the game drove the Web 3.0 developers to build such a new infrastructure of the Web, which is often called the Semantic Web Stack or the Semantic Web Layers (Horrocks, Parsia, Patel-Schneider, & Hendler, 2005).[22] Berners-Lee et al. (2001, p. 43) summarise the purpose and the technical aspects of the architecture as follows.

> The Semantic Web, in naming every concept simply by a URI, lets anyone express new concepts that they invent with minimal effort. Its unifying logical language will enable these concepts to be progressively linked into a global Web. This structure will open up the knowledge and workings of humankind to meaningful analysis by software agents, providing a new class of tools by which we can live, work and learn together.

A visual summary of this succinct paragraph can be demonstrated in Figure A.1.

The Semantic Web architecture consists of five main stacks: representation, reasoning, query, trust, and interaction. The Representation stack consists of various applications that represent data in a structured way. The Reasoning stack provides semantically operatable data by introducing semantic markups. The Query stack provides interrogations on the web constructed in the reasoning stack to make inferences on the web. The Trust stack provides justifications of the inferences made, creates a trustable setting for conducting transactions, and

[21] Let us illustrate this with an outdated example, in any case. Before the time of first attempts of Web 3.0, there were myriad of pages, say about Middle East Technical University. One page exhibited the number of enrolled students, another stated administration details, another listed the upcoming events. What if we wanted to learn the current president? The only thing we could do was to visit webpages one by one since no search engine at that time could answer. Today, a Google search immediately shows it. What if we wanted to know the third president? Unfortunately, even today, we cannot get this information directly, yet the related pages are more or less displayed.

[22] Although there are various versions of the stack, we use the basic version in Berners-Lee and Swick (2006).

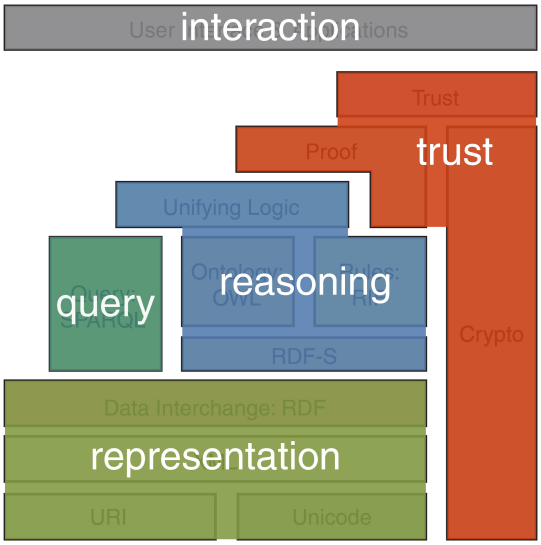


Figure A.1: A summary of the Semantic Web Stack

supports security and privacy. Lastly, the Interaction stack provides semantic technologies to the user.

The detailed visual representation of Figure A.1 is seen in Figure A.2. In the following, we will examine each stack in terms of its layers, respectively.

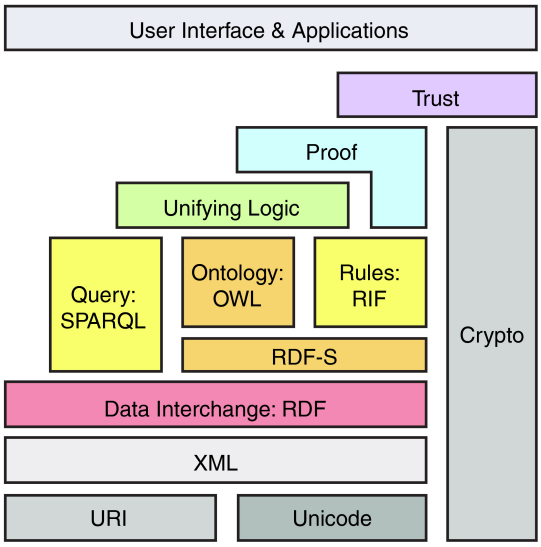


Figure A.2: The Semantic Web Stack

It is important to note that there are plenty of different web technologies. The Semantic Web Stack shows only the ones that are *recommended* by the World Wide Web Consortium (W3C).

A.3.3.1 The Representation Stack

The universal codes of HTML determine the webpages' format: Once we have such a format, we can fill in the contents. The lowest layer of the Semantic Web includes the basis of the hypertext technology, that of Web 1.0 (p. 285). In this layer, there are Unicode, URIs/IRIs. Unicode is the character set that is a computing standard for the encoding symbols. The reason for the inclusion of Unicode is to allow different natural languages to be represented and manipulated. URIs, Uniform Resource Identifiers, identify the resources on the Internet uniquely. In contemporary semantic web technology, developers use IRIs, Internationalised Resource Identifiers, generalization of URI, whose character set is extended to the Universal Coded Character Set (Keil, 2016). Thus, this layer is the basis upon which the whole Semantic Web will be constructed.

The second-lowest layer was borrowed from the existing hypertext technologies of that day that allowed the developers of Semantic Web to touch upon the content. The eXtensible Markup Language (XML) is a markup language that provides flexibility that users not only create their own labels on the content and create structured web documents, but also their own markup language (Berners-Lee & Fischetti, 2001). Suppose that one, say Laura, wants to create a CV webpage in XML syntax.[23] She has no explicit way of representing the notion of her name or her address: She needs to find a way to represent entities in the world on a webpage in a machine-readable format so that the content would be accessible throughout the Web. She can create new tags, just like the ones in HTML. For example, a header is specified as `<h1> My CV <\h1>`, this format of the header is standardised by the `<h1>`, tag. Now she can add her name as `<myname>Laura<\myname>`. So, not only the format but also the content is tagged, and HTML has been expanded by XML. In this respect, XML can be considered a data description tool, a tool for exchanging data over the Web.

[23] That is an XHTML page.

Moreover, this layer contains XML Schema. A schema is basically about the information that is expressed in the related system. Thus, XML Schema is a description for *an* application by restricting the structure of XML documents. It offers standardization of XML formats, which eases the data exchange.[24] For instance, in the CV example, XML Schema defines the structure of the CV by defining entities and attributes, their data types, and default and fixed values for entities and attributes (W3Schools, 2019). To sum up, the XML layer is responsible for the exchange of *structured* data over the Web by its virtue of providing a syntax.

So far, the Web contains a bunch of distributed structured data from multiple sources that need to be federated. Another technology is needed that glues those tags and a set of rules that makes *semantic* linkages that can classify the tags; so that the Web would end up with a global way of identifying entities as the global web. To this end, the developers use the Resource Description Framework (RDF) that is the third layer of the stack. RDF is basically a data model that accesses, connects, and describes the resources by introducing them in a set of triples (subject, predicate/verb, object) to encode the meaning.

Built on top of XML and connecting the tagged resources in a directed way, RDF has the graph structure: the subjects and objects are the nodes, and the relations between them are edges. This structure allows a consistent way of representing distributed data in a single model: Any constituent of a triple with the same IRIs come together from multiple sources to form a global source. As is, RDF is a data model for metadata that provides a syntax so that the distributed data from multiple sources can merge. Thus, stored in the RDF triples, the custom-built tagged words are now readable by the machine, which entails that the collections of triples form a web of data with the unique relational paths among the tagged words in the webpages (Berners-Lee et al., 2001, p. 40).

[24] Recall that there are several XML schema languages other than XML Schema, or there are other markup languages or ontologies other than the ones in the layers of the Semantic Web Stack. People can use other technologies, yet those are just recommendations of the World Wide Web Consortium.

A.3.3.2 The Reasoning Stack

XML and XML Schema cannot place semantic restrictions over either the standardised or the new labels defined by content producers. For instance, Laura can label her full name as `<name>Dr. Laura Phaenarete<\name>` or as the follows:

```
<name>
      <firstname>Laura</firstname>
      <lastname>Phaenarete</lastname>
      <title>Dr.</title>
</name>
```

In the real world, the first representation of the person's name refers to the same person to which the second representation refers. In the realm of the Web, on the other hand, these two representations take different URIs. Alternatively, consider this: (1) the planet Venus, (2) the Morning Star, (3) Lucifer, (4) Phosphorus, and (5) John Wycliffe have different URIs. The Web infrastructure has no way of knowing that in some contents (1), (2) and (4), in other contents (2) and (3), and in others (2) and (5) have to be treated as the same entity. Moreover, both (1) and (5) refers to (2). The former is a planet, and the latter is a human. Since RDF aims to connect resources in a consistent way to reach a single data model that allows inferences, it has nothing to do with providing any semantic differences.

The first layer of this stack is The Resource Description Framework Schema (RDFS). RDFS was developed to solve such problems. Like other schema languages, RDFS specifies how the resources and relations in RDF should be formalized. It describes a vocabulary that describes the notions of commonality and variability by founding its syntax in set theory. In other words, RDFS vocabulary, such as classes, subclasses, properties, sub-properties, domain, range, is used to describe the RDF triples.

RDFS applies some inference rules by relating the resources in classes and subclasses, domains and ranges, properties, and sub-properties. For instance, a first name and a surname construct a person's name: When 'name' is taken as a set, then 'firstname' and 'lastname' be elements of it, and such a structure must

belong to a person. Thus, RDFS relates these three resources in a set-theoretic approach to make inferences. In the latter example, once the classes of (1) and (5) belong is identified by RDFS, there would be no semantic clashes in the system. Hence, RDFS leverages the inference power.

Hitherto the Web gained powerful expressiveness by XML and RDF technologies and an inference system with RDFS. As mentioned, the semantic differences cannot be obtained by these technologies. The second layer of the Reasoning stack, OWL Ontologies, realizes semantically operatable tagged data. In this section, we firstly investigate the ontologies in information sciences and then speak of OWL.

Leaving aside its philosophical connotations,[25] an ontology in information sciences is a knowledge representation tool. An ontology uses a domain's entities and their interrelations, then provides classes, instances, properties, axioms, and definitions of the domain.[26] All the information about an entity of the domain at hand can be explicitly represented in the semantically enriched web. Thus, ontologies leverage the semantic aspects of data.

Apart from Web 3.0, ontologies can be built for several reasons, such as creating a knowledgebase of a domain, checking the consistency of the representation, or making inferences. That is, ontologies are not confined to use on web technologies. There are many languages to implement ontologies, OIL, LOOM, and KIF to name some. Out of them, W3C recommends the Web Ontology Language (OWL) to represent web contents,[27] so that a unifying logical language enables concepts to yield a linked global web. Built on RDF and RDFS, OWL representations are run to verify contents' consistency and deduce information from them. In other words, OWL enables the definitions of new classes and properties over the existing classes and properties defined by RDF and RDFS for modeling more restrictions to make valid inferences. Consequently, the static nature of

[25] A philosophical and computational journey of ontologies is elaborated in appendix Ontology.

[26] See section Ontology.

[27] OWL is not limited to use on the Web either.

Web 1.0 and Web 2.0 was limited to human readability; alas, Web 3.0 evolved into a 'meaningful' readable Web with the integration of OWL Ontologies.

Let us concrete this technical process with an example. Suppose that we are researching a person whose personal data distributed to various webpages and that we want to find the postcode of her home address.[28] In a static webpage, there is nothing like a 'postcode' tag, so it is cumbersome to look at all the webpages she has. In Web 3.0, all those webpages are connected by that person's name, identified by a specific IRI, and we hope to reach the postcode easily. It can be tagged as `<postcode>06800</postcode>`. Thus the machine is thought to reach this data. However, what if we are looking for the postcode of her home address. In such a situation, tagging cannot provide a solution. As stated before, this data was encoded in RDF, so it is related to other data; then, the machine can detect the postcode linked to the home address. Nevertheless, people use different terms to refer to very same things, so do webpages. For instance, one uses 'zip code' instead of 'postcode,' another uses 'postal code.' At this point, ontologies are used for figuring out the semantic content of the data, and their sets of inference rules are applied to provide accuracy of Web searches and to relate the data (Berners-Lee et al., 2001, p. 41). XML allows users to add arbitrary tags to their content, but RDF cannot fully express the meanings of those tags. The meaning of the whole thing is rooted in ontology. Web clients can get the equivalence of distinct terms or the XML tags by virtue of ontologies. Thus, it is ontology's role to serve a basis upon which the web client gets that those three expressions refer to the same number. Therefore, an OWL ontology, a semantic markup, enables to derive of relevant connections.

The third layer of the Reasoning stack is the Rule Interchange Format (RIF). RIF was created to enable the interchange of rules over the Web. Interchanging rules gains importance when different information models are merged. Suppose that there are two diverse information models. Each has its own rule sets that are expressed in different rule languages or dialects. Creating common represen-

[28] This example is developed from Berners-Lee et al. (2001, pp. 40–41).

tations, ontologies simplifies access to diverse models, but they are not capable of expressing mappings between the rules of these models. To merge them, we need to merge the rules that are also expressed for different purposes. RIF helps us bridge the rules with semantically preserving mappings. Thus, RIF, the common rule representation, enables the gradual development of a network of mappings between the rule sets of myriad information models in the Web.

The Reasoning stack is established for Web-scale reasoning. In most cases, reasoning on the Web is carried out by logic since checking the validity of inferred results is of the most importance. However, traditional reasoning methods are not compatible with Web-scale systems. The number of axioms and facts annotated on the Web is overgrowing. This makes the Web an open system whose completeness is not realizable. Besides, due that the inference capabilities of the formal systems differ, the users can prefer to model their data with different conceptual schemas. The Web contents are created by humans who have conflicting ideas, which makes the system inconsistent as well. The reasoning tools, on the other hand, are based on logic, mainly on the first-order logic, or its fragments, which are to deal with computational resources and time limitations. There are various reasoning engines[29] constructed upon a formal system, whose interoperability is problematic. In order to minimize these difficulties, Unifying Logic is introduced as the last layer of this stack.[30]

The most often used logics in semantic web technologies are description logics and Horn logics. Nevertheless, there is no single unifying logic recommended by W3C. When we have it, representations, axioms, queries would be expressed by the same formal system. It would be easy to sustain the usability of inferred results and relate them with the requested resources. In light of this, the Unifying Logic layer is above the Query stack since queries are expressed in some formal language. In other words, although we have a functioning Query Stack, we are still trying to find a unifying logical framework that provides formal semantics for making inferences.

[29] See section Ontology.

[30] This layer becomes more meaningful in the following lines.

A.3.3.3 The Query Stack

The Reasoning Stack, apart from the Unifying Logic layer, provides machine-readable and semantically weaved Web. The Query Stack enables making inferences on this system. The W3C recommends SPARQL, SPARQL Protocol and RDF Query Language, a semantic query language on knowledgebases for reprieving the data in the RDF format. There are ongoing developments in SPARQL to support OWL semantics fully.

A.3.3.4 The Trust Stack

The Trust Stack is being constructed for assuring confidence in reasoning and in using the Web. Above the Reasoning Layer, Web 3.0 should ensure to provide human-understandable explanations of the chain of reasoning. Further, it should also ensure privacy, security, and protection of digital rights in each layer.

The Trust Stack consists of two horizontal and one vertical layer. Firstly, the Proof layer is for manifesting/representing the inference process. In other words, it executes the rules provided from the Unifying Logic layer and provides justifications/ explanations of the inferences. Secondly, the Trust layer tells us whether we should count on the given justification/proof or not. The formal system could work perfectly, but the results could be unsound. Lastly, the vertical layer, Crypto, saves us from misleading conclusions: The Digital Signatures provide the proof that the RDF triples, OWL constructions, the selection of ontology vocabulary, the methods of proofs are written or constructed by a confident person or a confident institution. Since the signatures are based on cryptography, the trustworthiness of these people or these institutions cannot be affected. Thus, the applications on this layer make use of the sources with digital signatures to ensure the level of trust and support privacy and security. This stack is under development to provide valid and sound inferences from the secure Web.

A.3.3.5 The Interaction Stack

The Interaction Stack contains developing suited user interfaces and applications that enrich the Web semantically. To extend the usage of Semantic Web technologies, researchers aim at designing user interfaces for people who have almost no logical background. Moreover, they are developing applications for searching semantic data, enhancing search results by metadata, easing data integration from various sources, and enhancing knowledge management in web-based systems. As the preceding stack, this stack is still evolving.

A.3.3.6 Summary and Final Remarks

To sum up, the up-to-a-great-degree-not-realized layers of the Semantic Web stack are the essential components of Web 3.0 since only these layers could provide machine judgments over machine-readable data. Web 3.0 lacks an overarching logic, which could provide rules for making inferences in many contexts; an automatic proof checking system could explain how the machine reached that conclusion and whether the sources and the conclusion were trustworthy. Hence, these drawbacks disconfirm that Web 3.0 is the Semantic Web.

On the Semantic Web path, the initial aim was to connect and consistently integrate data. To realise this, data must be in a machine-readable format, and meanwhile it must be flexibly tagged by the content producers. Unicode allows every natural language to be expressible on the Web; XML provides an infrastructure for tagging the content freely by its users. Once the content is in the machine-readable format, it is now time to introduce the semantics on the grounds of inference. However, the tagged resources are spread all over the Web; there must be a way to connect them in a meaningful way. RDF provides a distributable and machine-readable model of the data: Resources are connected to each other as triples. The triples also should be brought together, otherwise the Web would be full of triples without signifying a meaning. The resources with the same URIs are connected to each other, which leads to a connected data collection. RDF reveals meaning of the resources through triples, yet the meaning in broader sense, namely the meaning of data which comes from

multiple resources, to a degree, is provided by RDFS which supports inference mechanism by defining rules over RDF. It expresses meaning through specifying inferences by assigning set-theoretical structures over the triples. As RDFS is built upon RDF in order to increase logical constrains, accordingly to increase inference power, OWL is built upon RDF due to the very same reasons. Data is, then, modelled with constrains between resources, classes, and properties: OWL sets further rules for describing classes based on allowed values for properties to empower the inference and check the validity of the inferences. In the end, we get a global web of connected and integrated data with valid inferences. Should the Web is a collection of links, the Semantic Web is the collection of linked meaningful data that the machine can read to process data integration, automation of some tasks, and above all, data discovery.

Web 3.0 becomes a worldwide database that has high data richness. The web browsers became capable of semantic- and context-aware searches, and even in some platforms, there can be inferences on the Web. All these revolutionary aspects of the new Web cannot swipe some deficiencies that the worldwide database has: the data is heterogeneous, there are inconsistent facts, the Web is open and growing rapidly. Nevertheless, the Semantic Web dreams that the machine can understand and respond to human questions nested in semantics. When this dream comes true, they will be producing knowledge by virtue of the Web that has been woven for decades. "Link by link we build paths of understanding across the web of humanity" says the inventor of the Web (Berners-Lee & Fischetti, 2001, p. 204), by embracing all the difficulties we have and will face, however. For instance, the fact that different ontologies are built upon different axioms and/or logics can be the reason for logical contradictions. This causes invalid results that can be hard to detect.[31]

[31] When we affirm the consequences, one logical form we use is *modus ponens*, which can be stated in the form that "If p then q, and p. Therefore q." An argumentation in the form of 'If p then q, and q. Therefore p." is rendered as deductively invalid since there may be other reasons for the existence of *q*. On the other hand, as Walton (2014, p. 143) states in his Measles Inference, "If a patient has red spots (of a certain kind), then the patient has measles, and this patient has red spots (of this certain kind). Therefore, this patient has measles." Although this inference is deductively invalid according to a logical system that includes *modus ponens*, the inferences like measles inference should be introduced to logical systems since the only way of having measles is having a certain kind of red spots. Thus, two ontologies, one strictly having *modus ponens* invalidates the measles inference,

We have to acknowledge the vast number of webpages and, accordingly, of the contents we have and a considerable amount of fallacious content on the webpages.[32] This fact troubles trustworthy semantic endeavors in the Web. Above all, the alleged semantic web cannot lead to discoveries yet. Web 3.0 defines structured data: data is stored in an organized way that warrants automation, integration, inferences, and integration with data analytics that promise prediction, personalized web experience, and alike. On the other hand, in the age of Big Data, what we have is unstructured data that nonuples the amount of structured and semi-structured data we have. To this respect, it would be illegitimate to say that we have *the Semantic Web* in Web 3.0, as Boden (2016, p. 38) clearly states:

> The semantic web isn't the same as the World Wide Web—which we've had since the 1990s. For the semantic web isn't even state of the art: it's state of the future. If and when it exists, the machine-driven associative search will be improved and supplemented by machine understanding. That will enable apps and browsers to access information from anywhere on the Internet and to integrate different items sensibly in reasoning about questions. That's a tall order.

Evolution to the *real* semantic web –at least the one Boden utters– may require a revolution in our understanding of data and its computation, which eventually can make the future closer, and the tall order simpler.

have two different inference models.
Besides, consider logical systems, such as intuitionistic logic, which rejects the law of excluded middle. Whether there is a unifying logic that accepts this law in some situations and rejects it in others is a question of debate.

[32] As of May 26, 2022, there are at least 4.87 billion indexed webpages. See WorldWideWebSize.com.

B. ONTOLOGY

In Web 3.0, we spoke of ontologies in the Reasoning Stack of the Semantic Web Stack. Recall that ontology is the backbone of Web 3.0 since it offers semantically processable data. In other words, the ontology technology paved the way for structured 'meaningful' readable data and reasoning in the Web, without which Web 3.0 could not have occurred. As mentioned in the appendix chapter From 1.0 to 3.0: Industry, Science, Web, ontologies are not confined to use on web technologies. They are crucial in information systems for their indispensable contributions to knowledge representation, knowledge management, knowledge sharing, knowledge production, and the like. Recall also that knowledge representation, knowledge management, and knowledge production are critical issues in Science 3.0. Ontologies must play a crucial role in knowledge management, which is essential for Web, Science, and Industry. For this reason, we believe it is worth examining ontologies in a separate appendix chapter, propounded for the four-phase picture of Industry, Science, Web, Data, and Ontology.

Ontology is one of the oldest human endeavors, along with science. It, as its etymology offers,[1] refers to the study of being. This subdisciple of philosophy has been answering the two-millennia-old question of what there is, if there is. Philosophers apply ontological methods to convey analyses on ontology's fundamental concepts, such as being and existence, concreteness and abstractness, identity and essence, universality and particularity, actuality and potentiality, part and whole, subject and property, attributes, and relations. Upon their reflections, each philosopher has their own ontological doctrine.

[1] Ontology comes from the Greek word ὄν, "being," that is the present participle of εἰμί, "to be." When λογία is added to ὄν, the word ontology is constructed as the study of being.

Ontology as a philosophical activity is classified in various forms. Strawson offers two kinds of ontologies in his *Individuals: An Essay in Descriptive Metaphysics*: descriptive and revisionary. "Descriptive metaphysics is content to describe the actual structure of *our thought* about the world," and revisionary metaphysics, which is "at the service of descriptive metaphysics," is "concerned to produce a better structure" for *our thought* about the world (Strawson, 2002, p. 9) [emphasis added]. The character of both metaphysics is to introduce an account of "thought structures" (Phillips, 1967, p. 105). That is to say, the discussion between descriptive and revisionary ontologies is rooted in analytic philosophy tradition.

In other respects, Keinänen (2008, p. 24) defines two kinds of ontological activity. The first one is the "modeling of our description of the world by means of ontological categories." In this kind of activity, the world's true descriptions are presented in some conceptual scheme and formalized -usually- in predicate logic. Whereas the second kind of ontological activity concerns the "direct characterization of the structure of the world in terms of ontological categories." Keinänen urges us that both descriptive and revisionary metaphysics are activities of this kind. He states that there is a kind of formal ontology which can be regarded as "the confluence between a school of thought which has addressed metaphysical problems within the mainstream of analytic philosophy, and another school more closely related to phenomenology." A fortiori, Guarino (1995, p. 628) defines the former school as a school consisting of philosophers who agree on the idea of "descriptive metaphysics" proposed by Strawson, and the latter school as a school consisting of philosophers who follow the tradition of Brentano and Husserl.

Poli (2003, p. 185), on the other hand, defines three forms of ontologies: descriptive, formal, and formalized. Descriptive ontology is about organizing *prima facie* information; formal ontology "distills, filters, codifies and organizes the results of descriptive ontology"; formalized ontology, lastly, is formally codified constructions that "descriptively acquired" and "formally purified." This version of the classification of ontology is the closest to our purposes. Descriptive ontology, Ontology 1.0, encapsulates the traditional ontologies that aim to de-

fine reality by classifying *all* types of entities in all spheres of being. Thereby most of the time, these ontologies considered metaphysics. Poli mentions both Husserl and Quine are of formal ontologies (p. 186), and he separates formalized philosophical ontologies. To our understanding, both formal and formalized ontologies pave the way for ontologies in information systems, so the latter two forms of ontologies should be merged. Thus, formal ontology *and* formalized ontology, Ontology 2.0, captures theories that are developed, expressed, and refined with formal tools, such as algebra, category theory, mereology, set theory (B. Smith, 2014, p. 77). However, constructed for computational purposes, Poli's formalized ontologies are in the realm of Ontology 3.0. These ontologies are built as software artifacts for computational purposes, which is at the heart of this appendix chapter.

B.1 Ontology 1.0

Ontology 1.0 refers to the philosophical study of being, which can be called traditional ontology. Ontology 1.0 studies which entities deserve to be called "being" and its reasons; the criteria of entities that must be labelled as "being," which are the relations between those beings and alike. There are myriad answers to those issues, so as various kinds of ontologies. Accordingly, the work of a philosopher who studies traditional ontology,[2] is to illuminate or detect what there is and to decide properties of and kinds of relations among those entities. That is to say; a traditional philontologist develops a hierarchical and relational web of beings. Among traditional philontologists, we choose Parmenides, Aristotle, and Whitehead for providing different aspects of *reality* construction.

Parmenides of Elea is famous for denying the existence of change. He starts his argumentation with that only being *is*, and non-being *is not*, thereby unthinkable cannot be an object of thought (Kranz, 1994, p. 81). Upon this argument, Parmenides builds his philontology: Being is unchanging, indivisible, knowable,

[2] From now on we call "philontologist" to a philosopher who studies ontology; and "philontology" to philosophical ontology.

ubiquitous, self-identical, and eternal. There *seems* change, generation and destruction, and process; the reason behind these illusions is our senses, in which we should not trust (Zeller, 2008, p. 82). Moreover, our senses betray us by making us think of contraries. Then, there is no dichotomy at all: Even if there seem contraries, one of them is real, and it *is*. So, non-being, contrary to being, cannot be an object of thought, since what is in thought *is*. Thus, in Parmenidean ontology, there is only 'One' that we can speak of; consequently, there is no room for explaining velocity as a fact.

Aristotle constructs his ontology, the science of being qua being in the *Metaphysics*.[3] Throughout the book, he examines what is worthy of being called 'being,' which has several senses: Accidental being, being in the sense of true being, being of the categories, and potential and actual being (Cf. Brentano, 1975). The first sense cannot be at the top of the taxonomy since an accidental being can cease existence. The second sense is not in the ontology realm since a true being is only in thought, just like non-being is a false-being. The third sense, on the other hand, gives us the hierarchy of being in the tangible world. In *Categories*, Aristotle speaks of primary categories as substances and secondary categories as attributes or categories. Secondary categories cannot be the ultimate study of ontology since their existence depends on the primary categories, namely substances.

> [S]ome things are said to be because they are substances, others because they are affections of substance, others because they are a process towards substance, or destructions or privations or qualities of substance, or productive or generative of substance, or of things which are relative to substance, or negations of one of these things of substance itself. (1003b5-10)

Substances, says Aristotle, are said in different ways. In 1028b33-35, four senses of substance are given as substratum, essence, universal, and genus. The last two candidates are ruled out from the investigation of being, for they cannot satisfy the fundamental requirement of being a substance, which is separateness.[4] The examination of whether substratum or essence merits to be called substance, on the other hand, tells that both senses can be reduced to form. At this point, the

[3] We use the Aristotelian corpus edited by Barnes et al. (1995).

[4] This section is prepared for giving a general idea of traditional philontologies. If one wants to learn more about how we ended up this conclusion, they can read Z, H, Θ in the *Metaphysics*.

fourth sense of being, actual and potential being, must be apprehended. What merits to be called 'being' in Aristotle's ontological hierarchy is *form*, which is actual, separate, and individual.[5]. In sum, being under the analysis of actuality and potentiality is the form; the hylomorphic analysis[6] reveals that substance is the bearer of the attributes. In other words, combining his other doctrines, Aristotle ends up with a substance in the highest sense as the highest being is actual, separate, and individual form. It seems that Aristotelian ontology encapsulates all the beings from sands to stars and provides a place for velocity, yet there is no room for writing a dissertation as a process in this ontology.

Whitehead is known as a process philosopher,[7] and remarkable mathematician. A process ontology develops through process metaphysics, says Whitehead, and this metaphysical attitude replaces material objects with momentary events of experience (Whitehead, 1957). This doctrine is against most philontologists in Ontology 1.0, for the dominant view in Ontology 1.0 is metaphysics of substance/objects. Whitehead, on the other hand, deprives inert substances of actuality,[8] and claims that the substances belong to space-time and are related to each other externally. Then, he puts actuality into momentary events,[9] the events that are *actual entities* or *actual occasions*: Being internally related with each other, the events are "the final real things of which the world is made up" (Whitehead (1957, p. 18), Cf. Ford (1983)). Naming things that exist as 'organisms,' Whitehead claims that organisms are not only internally related but also externally, adding that they are interdependent and intrinsically active. To sum up, everything in the world can be referred to as some actual entity. Thereby Whiteheadian ontology is not a hierarchy, rather a web of entities and the relations between them, in which we can find writing a dissertation as an entity.

[5] See *Metaphysics*, Λ.

[6] Hylomorphic analysis tells the degree of the matter-form constitution, and accordingly of potentiality-actuality of a substance. So, all the sensible objects are subject to hylomorphic analysis.

[7] Indeed, the godfather of the process philosophy is Heraclitus. Compare the process philosophy with Parmenidean, one who denies the existence of change.

[8] Recall that actuality is the most characteristic of being in Aristotle, like in most, if not all, philosophers.

[9] Since, action and passion are always simultaneous.

Ontology 1.0 is the traditional philontology that encompasses different examinations of how humans depict the world. Philontologists question whether numbers exist, whether there are things called processes, what the ultimate being is, and so on, and give different answers for centuries. That what one says there is, another says not triggers us to think about how we are supposed to depict the world in the machine.

B.2 Ontology 2.0

Ontology 2.0 refers to both analytic ontologies and formal philontologies.[10] It involves, in the broadest sense, philontologies that aim at striping contemplative, say revisionary, a feature of philontologies and at striving to reach objectivity, as highest as possible, towards reality. To elucidate Ontology 2.0, we will introduce Husserlian formal ontology to a degree since its pivotal role not only in Ontology 2.0 but also in Ontology 3.0 should be grasped. Next, we will speak of Quinean (analytic) ontology that shares Husserlian formal ontology's same pivotal destiny. Then, we will state the reasons for naming these two philosophers under the same aspect –for some scholars claim that they belong to the worlds apart– by examining and defining formal ontologies.

B.2.1 Husserlian Ontology 2.0

Being worth mentioning as the first philosopher of Ontology 2.0 belongs to the coiner of the term 'formal ontology': Husserl. A narrative of his general philontological endeavor can be begun by introducing his ramification of types of objects: Fact, Essence, and Meaning (D. W. Smith, 2007, p. 157). The realm of Fact includes real individuals that exist in space and time, which can be grouped into concrete individuals (books, cats, stars), state of affairs (Socrates' being wise, Romeo's loving Juliet), and events/processes (writing a dissertation,

[10] Consider that Ontology 2.0 refers to the philosophical endeavor of formal ontologies, while Ontology 3.0 is not in the realm of philosophy, albeit referring to formal ontologies.

flood) (*Ideas I*, §2). Under the realm of Essence, there are eidos[11] of objects. These objects, not in space or time, determine concrete-spatiotemporal objects; species, qualities, unity, plurality, and whole and part, to name a few (*Ideas I*, §§ 10-12). The objects in the former realm are particulars with contingent characteristics, whereas the ones in the latter are generalizable, unchaining, and timeless (*Ideas I*, §§ 6, 8). The last realm is Meaning/Sense that embraces objects that are the ideal content of intentional experience, for meanings/senses have a distinguishable role in intentionality, and that role deserves to be discussed in such a categorical ontology. As D. W. Smith (2007, p. 156) points out, Husserl does not speak of this last realm explicitly, though it is essential for his ultimate construction, phenomenology.

In sum, Husserl lists both categories of concepts/meanings, as Kant does, and categories of objects, as Aristotle does. He bonds these two with *semantic correlations* (which can be found in Frege's philosophy); then he ends up with a categorization of concepts/meanings and the objects they represent, and of our experiences, those are "intentionally related to objects via meanings that represent such objects" (D. W. Smith, 2007, p. 138).

Within the design of this dissertation, if we are to speak of Husserlian ontology, we must dig out the realm of Essence, in which two kinds of essences appear, those pave the way for material and formal ontologies. The first kind of essences is called material essences that are generalizations of the realm of Fact. The highest genus of material essences is called a *region* (*Ideas I*, § 9). In the words of Husserl, a region is "the total highest genetic unity belonging to a concretum, i.e., the essentially unitary nexus of the summa genera pertaining to infimae species within the concretum" (*Ideas I*, § 16). Husserl purports the subtle work of determining the number and distinguishable features of the regions due to the different regions' interwovenness (*Ideas I*, §152). Despite this fact, D. W. Smith (2007, p. 157) suggests that there are mainly three regions: Nature, Consciousness, and Culture or Spirit. Nature subsumes all entities in nature; Conscious-

[11] Husserl uses "eidos" in the sense of Platonic Forms.

ness subsumes all conscious experiences; Culture subsumes all entities formed by human acts. Put differently, the material essences in the first-mentioned region concern the structure of time, material compositions, and causality; the ones in the second concern the structure of intentionality, and the ones in the last region concern acts of people in communities (ibid., pp. 184–185).

Each material essence is studied by a corresponding regional/material ontology.[12] Accordingly, regional ontologies apply to all objects whatsoever, since "to the pure regional essence [...] there corresponds a regional ontology" (*Ideas I*, §9). For instance, the regional ontology of Nature is a hierarchy of genus and species are determined by ordering eidetic singularities, which are the individuals in nature. Besides, each region has its own set of a priori truths;[13] those apply to any possible categories in that region; viz, a regional ontology confirms a set of regional categories: It is the regional essence that "makes up the content of the regional ontology" (*Ideas I*, §16). So, the regional categories are eidetic universals and apply to individual objects in that regional ontology. For instance, *Ideas I*, § 9 says Nature, as a highest regional essence, corresponds to "the eidetic science of any physical Nature whatever"; that is nothing but the ontology of Nature. It, therefore, explicates the categories of empirical sciences of Nature "with rational purity" in order to theorize the grounds of this empirical science.

The second kind of essences are formal essences that are materially empty, or purely logical (*Ideas I*, § 12). Formal essences, in other words, are the forms of objects of any type. The highest genus of formal essences is *category*, just as it is *region* for the material essences. D. W. Smith (2007) explicitly puts that Husserl offers different lists of categories (p. 145) and mentions "there is always more to come" in Husserl's categories (p. 156). For instance, the categories found in *Ideas I*: individual or substrate, species, quality or property, relation, state of affairs, connection, necessity, possibility, dependence, independence, whole, part, unity, plurality, number, set, group, manifold, value (p. 157).

[12] In the context of material essences, the adjectives of 'regional' and 'material' are used interchangeably.

[13] "Each regional essence determines 'synthetical' eidetic truths, that is to say, truths that are grounded in it as this genetic essence" (*Ideas I*, §16).

The regions are strictly separate from each other; for instance, the ontology of Nature is different from the ontology of Culture since each region has its own syntactic a priori truths that govern their sub-ontologies. That is not the case for categories: A formal singular essence can lie inside another essence, even a higher essence can lie inside a lower one (*Ideas I*, §12). That is why categories can be defined as the "genus of all genera" (*Ideas I*, §§12-13). This non-hierarchical feature of categories, which cannot be found in material ontologies, enables them to be applicable in any material domain. In other words, formal ontology can be applied, for instance, to the ontology of Nature and the ontology of Culture and their sub-ontologies simultaneously. To wit, the charm and significance of formal essences come from the fact that the same formal essences apply to objects in any of these regions. Further, the gist of Husserlian formal ontology is that any object has a formal essence, and formal ontology is the eidetic science of any object whatsoever (*Ideas I*, §10).

Husserl distinguishes between regional ontologies, which study material essences shared by all the entities in the region, and formal ontology, which surveys the essence of objectivity in general.[14] In other words, formal ontology is the science of the objects *qua* they are objects. Further, formal essences strictly govern or determine all objects in any regional essences (D. W. Smith, 2007, p. 145). Therefore, the realm of Fact cannot be separated from the realm of Essence. Indeed, the sciences of the latter govern the sciences of the former. In *Ideas I*, §9, Husserl states empirical sciences depend on material ontologies. Thus, the role of formal ontology is governing or determining the structure of material ontologies, which constitutes empirical sciences; i.e., anything and everything is in the scope of formal ontology. Then, every empirical science has essential theoretical foundations in formal ontology (Cf. *Ideas I*, §2). A Husserlian formal philontologist's work is to detect and define the formal essences that govern all, or for most of, the objects. Then she can *copy* the formal essences then *paste* them into regions in order to reach material ontologies.

[14] Husserl neither gives a list of formal and non-formal entities nor provides a demarcation rule that distinguishes these two. Rather, he charges the philosophers to find out the objects and their essences, thereby enhancing the regional and formal ontologies. Cf. *Logical Investigations* I, §71.

Husserl uses the term 'formal,' in the broadest sense, as opposed to 'material' or as 'domain-neutral.' A formal theory, then, mentions any particular object, yet at the same time deals with those in total *abstracted* ways (Simons, 1982, pp. 113–114). In this respect, formal logic does not directly deal with the meaning of a particular proposition; instead, it discusses the eidetic essences of any proposition whatever. The same is true for formal ontology: it applies to all domains of objects whatsoever. footnoteMoreover, the demarcation of formal and non-formal is not related to being expressed symbolically (Simons, 1982). Axioms of mereology in the Husserlian sense, for instance, are not written in a formal language; rather, they are of no domain and consist of purely logical constants and formal concepts, such as "Two distinct objects cannot be part of each other." Note that this does not mean that we cannot express such axioms in a formal language. Any determinate proposition-form and any form of a proposition member is an eidetic singularity in pure logic, just like any shape, quality, or mental process being an eidetic singularity in formal ontology (*Ideas I*, §12). If the highest genus of pure logic is any signification whatever, which is, then the highest genus of formal ontology is categories which are of any object whatever. Hence, it is evident that formal ontology is not a formal (logical) language or pure logic itself; rather is a discipline totally distinct from formal logic, yet formal ontology is analogous to formal logic (B. Smith & Smith, 1995, p. 28).

The crucial point is that eidos or formal points out the non-material or domain-independent nature of this ontology, the same as formal logic. For Husserl, formal ontology is a counterpart of pure [formal] logic. Pure logic serves semantic correlations with logical forms, and formal ontology is "always as pure logic in its full extent as *mathesis universalis*" (*Ideas I*, §10; D. W. Smith (2007, p. 184)). To wit, formal denotes two fields in Husserl: ontology and logic. Ontologically formal deals with "whatever pertains to be object in general," whereas logically formal deals with logical operators and functors, and thereby logical forms behave like formal essences (Poli, 2003, p. 189).

B.2.2 Quinean Ontology 2.0

Quine is the second most influential philosopher both in Ontology 2.0 and Ontology 3.0. In his paper 'On What There Is,' Quine (1948, p. 21) starts his ontological analyses with the decipherment of the puzzlement of Plato's beard: "Nonbeing must in some sense be, otherwise what is it that there is not?" There are two alleged solutions to this puzzlement: Non-beings either have ideal-existence or are possible entities. The former entails that Pegasus exists as a reference to its ideal existence since Pegasus-idea does exist as a mental entity. On the other hand, states Quine, Pegasus, and Pegasus's idea cannot share the same ontological status: Pegasus' being a mental entity cannot entail its existence, besides it cannot provide any ontological condition to it. The latter alleged solution that nonbeings are possible entities puts forth that entities are ramified into actuals and possibles, thereby Pegasus *is* an *unactualized possible*, whereas concrete entities are actuals. Quine objects to this solution for two reasons. Classification of possibilities gets all balled up (then, possible *statements* should be opted over possible *entities*). Furthermore, the contradictory entities, such as "round square cupola", cause another ontological category to occur- unactualized *impossible*, which is obviously an oxymoron.

These objections could be defended by uttering that such phrases are meaningless. Nevertheless, meaning (that does not necessitate the existence of the term) and naming (that implies the existence of the term) are different things; for this reason, some shorthand descriptions ('the winged horse that was captured by Bellerophon') are taken as names ('Pegasus') (p. 26):

> When a statement of being or nonbeing is analyzed by Russell's theory of descriptions, it ceases to contain any expression which even purports to name the alleged entity whose being is in question, so that the meaningfulness of the statement no longer can be thought to presuppose that there be such an entity.

In light of Russell's theory, descriptive names can be transformed into variables of quantifications ('something,' 'nothing,' 'everything'); those variables by themselves are the bearers of meaning and existence. To construct on philontology,

we must then commit ourselves to positive existential statements.[15] This brings forth the famous Quinean ontology-construction motto(s): "To be is to be the value of a variable" (p. 34) / "To be is, purely and simply, to be the value of a variable" (p. 32).

However, this remark does not tell us what there is; instead, it only states what a statement in question says there is. That means building a philontology is like a scientific endeavor: Ontologists take entities from theories of natural sciences, then by using those, they build their philontologies (B. Smith, 2002, p. 17). When viewed in this light, building an ontology is an ongoing task; furthermore, it is discipline-related. Think of an ontologist who wants to construct an ontology of physics. She starts with using the entities that physicists are committed to, then represents them in the language of first-order logic (B. Smith, 2002, p. 18). Scientific methods in physics show the existence of these entities. Finally, this ontology can be merged with an ontology of, say, biology. In the case of choosing between two competing hypotheses, says Quine (1948, pp. 10–11), we should favor parsimony just like we do in natural sciences. In conclusion, Quinean ontology is extracting ontology from scientific theories: They reveal what there is, and ontology brings them together to form a network of entities and their relations through the medium of logic. Each natural science gives the entities to which it is committed, then each of them represents a partial ontology.

B.2.3 Ontology 2.0 in the 21^{st} Century

Cocchiarella (2007, p. xiii) defines formal ontology as:

> [...] is a discipline in which the formal methods of mathematical logic are combined with the intuitive, philosophical analyses and principles of ontology, where by ontology we mean the study and analysis of being qua being, including in particular the different categories of being and how those categories are connected with the nexus of predication in language, thought and reality. The purpose of formal ontology is to bring together the clarity, precision, and methodology of logical analyses on the one hand with the philosophical significance of ontological analyses on the other.

[15] Obviously, negative existential statements cannot commit us to an ontology. That is to say, existence must be positively quantified in order to reach an ontology.

In this respect, formal ontology is the study of all forms and modes of being expressed in a formal language. That would embrace Quine, for science uses formal language for its survival, and ontologies are the scientific endeavors. Nevertheless, Cocchiarella (2007, p. ix) demurs that Husserl studies categorical structures, not a formal ontology. The main reason for such exclusion is that Husserlian formal ontology is not rooted in quantifier-centered symbolic languages (B. Smith & Mulligan, 1983, p. 73; p. 74). B. Smith and Mulligan (1983) criticize analytic philosophers for ignoring the formal structures and relations among objects and kinds of objects. They state that "[t]he most powerful motivating force underlying the resistance of the analytic philosopher to the acceptance of an ontology of moments is his tendency to run together ontological questions with questions of logic or 'grammar'" (p. 79). Moments are elements or factors of wholes and are the indistinguishable concept of Husserlian formal ontology. What analytic philosophers do, for the most time, is translating sentences into logical forms, thus cannot room moments as logical constants. On the other hand, moments are not in the realm of formal logic, rather in formal ontology because they are about the reality of objects, not meanings. So, the perplexity of such objections from the analytic philosophy-side arises from the concept 'formal.'

As we noted earlier, a formal theory is domain-independent; if so, formal logic is a domain-independent formal theory that deals with structures or meanings. Whereas formal ontology is a domain-independent formal theory that deals with structures of objects and their parts (Cf. B. Smith & Mulligan, 1983, p. 73). Hence, formal logic concerns itself with neither objects, nor parts of objects, nor relations between objects, but sentences about the objects and their relations. Although Cocchiarella rules out Husserlian formal ontology for its lack of formal language, Poli keeps Husserlian formal ontology. Poli (1993, p. 1) mentions two distinct interpretations of the term 'formal ontology'. The first, he calls analytic, is the formal ontology in the sense of Cocchiarella: "[a] branch of ontology which is analyzed within the framework of formal logic." The second, he calls phenomenological, is the formal ontology in the sense of Husserl: "[explicating] both the connections between the formal and material, and those

between the ontological and the logical." Moreover, Poli stresses that "[d]espite their differences, these two varieties of formal ontology quite frequently overlap each other, although to date there has been no systematic study of the categories and layers that constitute formal ontology and no systematic analysis of the issues addressed by it." Beaney (2018) supports Poli's claim by stating that the analytic methods play a pivotal role for both analytic philosophy and phenomenology within the geist of the twentieth century.

Ontology 2.0 is distinguished from the previous philontologies by the analytic methods. [16] In the vein of Føllesdal (1996), analytic methods are characterized by two crucial elements of philosophical and scientific activities: arguments and justification. That is, the decompositional analysis of propositions, or in other words, conceptual analysis is just a *part* of the analytic method. Arguments are not restricted to deduction, rather include induction or abduction; justifications, then, include 'proving' as well as 'accepting' and 'believing.'[17] To wit, there are several approaches to reach sound arguments and valid justifications in analytic philosophy, which eventually shape the way of approaching ontology.

To sum up, within the elucidation of Ontology 2.0, we introduced two prominent philosophers: The former is the coiner of the term 'formal ontology' and developed his philosophy from an ontological perspective (B. Smith & Smith, 1995); the latter is famous for introducing criterion of ontological commitment, which can be regarded as the seal of analytic philosophy (Bricker, 2016). Then, we argued for the reciprocal influence of phenomenology and analytic philosophy. In order that we could purport that the methodology of Husserl and Quine are neither the same nor the one; yet we claim that both originated from formal-construction of ontology mindset. What is common in these ontologies is that formal theories are built upon formal foundations. Finally, what we should understand as formal ontology, as Ontology 2.0, today can be summarised by B. Smith (2002, p. 5) as follows:

[16] For more details on the relationship between analytic philosophical and phenomenological methods, see Beaney (2013). Moreover, Føllesdal (1996) stunningly discusses the history of philosophy into degrees of being analytical, which embraces continental philosophy, along with ancient philosophy, as well.

[17] See Føllesdal (1996) for the theory of reflective equilibrium as the approach of justification in analytic philosophy.

> [Phil]Ontologists nowadays have a choice of formal frameworks (deriving from formal logic, as well as from algebra, category theory, mereology, set theory, topology) in terms of which their theories can be formulated. These new formal tools allow philosophical ontologists to express intuitive principles and definitions in a clear and rigorous fashion, and they can allow also for the testing of theories for consistency and completeness through the application of the methods of formal semantics.

When ontologies are expressed in a formal language, then it becomes easier to represent the world to the machine. Ontologies were thought of beneficial to "the knowledge-construction process in yielding high-value knowledge bases" (Guarino, 1995, p. 626), and then on utilized in knowledge-acquisition, -integration, -sharing, and -reusing in many machine intelligence studies, such as knowledge representation, natural language processing, and theory of databases (Cf. Poli, 2003, p. 188). The following section will see how Ontology 2.0 is employed in computer science and information systems.

B.3 Ontology 3.0

In the previous section, we slightly mentioned that there are two kinds of formal ontologies. One is a philontology, studies being qua being with formal frameworks to test the theories for consistency and completeness; the other is a software designed for various purposes in information systems.[18] A philontologist, either of Ontology 1.0 or Ontology 2.0, aims at constructing the reality of what there is either partially or as a whole. She tries to elucidate the truths about the world within philosophical methods.

In contrast, an ontologist of Ontology 3.0 aims at designing a software agent. She is occupied with developing an ontology for specific purposes that are held

[18] Guarino (1998, p. 3) gives a list of fields that makes use of ontologies ("knowledge engineering, knowledge representation, qualitative modeling, language engineering, database design, information modeling, information integration, object-oriented analysis, information retrieval and extraction, knowledge management and organization, agent-based systems design") and a list of some application areas ("enterprise integration, natural language translation, medicine, mechanical engineering, standardization of product knowledge, electronic commerce, geographic information systems, legal information systems, biological information systems"). He uses the generic term 'information systems' to refer to fields and application areas en masse. We follow this.

in computational environments.[19] For this reason, such ontologies are also called *applied ontologies*.

The story of Ontology 3.0 starts from the introduction of catalog systems in library management. Books had been located in permanent shelves where the latest arrivals were located at the end of the shelves in early libraries. An increase of numbers and types of artifacts –such as books, periodicals, records– in libraries caused difficulty accessing materials both for librarians and users. In order to facilitate access to the materials, systematic classifications of the artifacts were needed. Classifications provided schemes according to which artifacts were arranged in a sequence that helps users find out the artifacts they were looking for. In 1876, Melvil Dewey introduced the Dewey Decimal System, a classification of documents in libraries that is one of the best-known schemes (Foskett, Estabrook, Francis, & Haider, n.d.).[20] Each artifact was categorized, tagged, stored, and retrieved by its unique identifier. From then on, the books have been located according to their unique identifiers that make it easy to find and return any book to its proper shelf.

The Dewey Decimal System was updated several times, and there introduced myriad classification schemes, such as the Library of Congress Classification, the Universal Decimal Classification. The common aspect of these different classification schemes is they all provide *standardization*. The libraries that use the same classification scheme locate their artifacts according to the same unique identifier. As long as the material's identifier is known, the material can be retrieved easily by deciphering it in *any* library that uses the same scheme. It can be viewed as the beginning of structuring data in information management before the advent of computers.

The philontologies as a classification tool in information systems after the advent of computers were firstly introduced by a computer scientist, George H. Mealy

[19] For the time being, we limit ontologies to specific domains.

[20] According to this scheme, artifacts were classified into ten groups, and each group was assigned a three-digit number, e.g. 100–199, philosophy and psychology and 500–599, natural sciences and mathematics; 600–699. These main groups were in turn subdivided over and over again for assigning more specific subject groups.

(B. Smith, 2002, p. 20). During his examination of the basic foundations of data modeling, Mealy used Quanian way of ontologizing into a part of his software system (Mealy (1967, p. 525), B. Smith (2002, pp. 20–21)). After philontologies got popularity in information systems, another prominent computer scientist, Gruber (1992, n.d.) famously articulated ontologies as a classification tool that provides standardization: "An ontology is a specification of a conceptualization." According to this definition, ontologies are designed for creating a controlled vocabulary of a domain. "A conceptualization," says Gruber, "is an abstract, simplified view of the world that we wish to present for some purpose" (ibid). Hence, an ontologist represents a part of the reality from a particular perspective by postulating an orthodox way. For instance, after this definition, a farm ontology created for an agriculture management system is designed to include the entities, their properties, and interrelations, where the terms for entities, properties, and relations are offered as a conceptualization.

Guarino, Oberle, and Staab (2009, p. 2) define ontology in information systems as "a formal, explicit specification of a shared conceptualization." According to them, the four constituents of ontology are indispensable. (1) The very distinguishing feature of Ontology 3.0 is that it is constructed for representing the phenomena into the machine, viz., an ontology must be machine-readable. (2) An ontology offers standardization. For instance, the following is a frequently encountered issue in information systems. A group of scientists uses some terminological and conceptual structures different from another group does. In this situation, a concept can have two different references, or a reference can have two distinct concepts, so as their formal expressions. For instance, Mosquito Gross Anatomy Ontology defines 'cell' as "an area of wing membrane delimited by veins or by veins and the wing margin."[21] Whereas, Cell Ontology offers the most used 'cell' definition as "A material entity of anatomical origin (part of or deriving from an organism) that has as its parts a maximally connected cell compartment surrounded by a plasma membrane."[22] these two entities nei-

[21] Term's IRI: $http://purl.obolibrary.org/obo/TGMA\_0000218$

[22] Term's IRI: $http://purl.obolibrary.org/obo/CL\_0000000$

ther in reality nor formally refer to the same thing. Such usages violate formal structures and cause invalid results. In order not to encounter such difficulties, an ontology should be a consensual construction. (3) The concepts, properties, relations, and constraints on these must be explicitly defined in formal means. For instance, a formal declaration of a circular definition of a property may cause a recursive loop that results in crashing the program. Notorious for its circular definitions, Friend of a Friend (FOAF), an experimental linked information system, defines 'image' as "a subclass of Document corresponding to those documents which are images."[23] In description logics, it can be represented as `Image = Document AND Image`. The only information one can derive from this definition is that Image is a subclass of Document. To wit, an ontologist has to provide clear formal definitions of the fundamental constituents of the related domains. (4) Conceptualization is an abstract model of some phenomena in the real world with identifications of the relevant concepts of those phenomena. That is, ontologies are real-world representations.

Hayes voices the understanding of the ontology of most of the people in information systems by uttering that the first thing to do in information systems is to "formalize the naive worldview, using whatever concepts seem best suited to that purpose" without caring philosophical worries of representing the world with "some special collection of concepts" (cited in Guarino (1995, p. 627)). On the other hand, the concept of 'conceptualization' is a controversial issue among ontologists. We believe that taking ontology as a particular way of conceptualizing is a notorious definition. As Guarino (1995) points out the modeling differences between the transfer view –modeling according to an expert's view– and the modeling view –modeling from the objective reality–, conceptualization can be taken as transfer view. Hence, for a better ontology, an ontologist has to appeal to the objective reality as it exists.[24] We will return philosophical aspects of ontologies soon, yet before it, let us give application areas of ontologies in information systems.

[23] `http://xmlns.com/foaf/spec/#term_Image`

[24] For the details see B. Smith (2004).

B.3.1 Ontologies as Software

In this part, we are going to explore areas of usage of applied ontologies, of which Semantic Web is an example.

Building an ontology, either in philosophy or in information systems, starts with identifying the entities, properties, and relations in the domains of discourse. The domain experts should share a domain's terminology, which is a controlled vocabulary, which paves the way for standardization in the domain. Thus, the first reason for building and using ontologies is to maintain a controlled vocabulary and express the terms in hierarchical structures and heterarchical networks. MeSH (Medical Subject Heading) terms, for instance, provides an exhaustive controlled vocabulary for indexing biomedical literature. Further, it indexes abstracts and/or citations of all documents, books, and references in PubMed[25], a comprehensive search engine that accesses vast databases on biomedical and life sciences.

Accordingly, ontologies are used for sharing and annotating domain knowledge. In parallel with, the domain experts use and/or expand the ontologies. GoodRelations[26] is one of the oldest controlled vocabulary used in e-commerce. It aims at maintaining interoperability between websites –which provide contents– and clients –who consume those websites. This ontology is used for search engine optimization, product information management, or e-commerce data quality management (Hepp, n.d.). Any industry for any kind of goods in any country or legal environment can use the information model of GoodRelations provides. The customers only need to adjust their special features into this ontology that serves as a domain knowledge template.

Further, ontologies play a crucial role in data integration and agents' interaction. As long as they are modeled via the same controlled vocabulary, many separate databases can be merged. That enables different data sources to become a uni-

[25] https://pubmed.ncbi.nlm.nih.gov
[26] http://www.heppnetz.de/projects/goodrelations

fied database so that the information expanded, just as agents' interactions. For instance, during the Human Genome Project,[27] each laboratory created and used its own terminology, classification, and semantic structure caused the problem of knowledge bases and/or databases not being shared among the researchers. The Gene Ontology[28] was introduced to researchers to solve that obstacle. Consequently, not only various data sources became a single thing, but also researchers have used the findings of other laboratories in their research questions.

Ontologies have a critical role in knowledge extraction; since they are not only offering a machine-readable controlled vocabulary but also semantically connected processable data. Recall from Web 3.0, the tagged data brought together by RDFS gains further semantic features with the introduction of ontologies. So, in general, ontologies are robust domain models for knowledge analysis. By knowledge analysis, we mean employing queries for accessing implicit knowledge from what is represented on the machine. Each new query is crafted to reach a result from the set of propositions presented in the formal system. For instance, in clinical information systems, testing and discovering a medical relationship or feature and examining its validity in the biomedical domain is the backbone of decision support systems (Yargan & Zambak, 2021). These operations are conveyed via queries to the related knowledge representations, namely to the related ontologies. In sum, beyond being standardizers of terminology and taxonomy, ontologies are tools that offer a logical structure upon which knowledge management systems can be built.

To sum up, applied ontologies are built for from the standardization of the knowledge to be represented in the machine to the knowledge management in fields such as science, industry, government, education, and healthcare. All said can be summarised by Poli (2003, p. 188):

> An ontologically grounded knowledge of the domain's objects should make their codification simpler, more transparent and more natural. Indeed, ontology can give greater robustness to models by furnishing criteria and categories with which to organize and construct them; and it is also provide contexts in which differ-

[27] https://www.genome.gov/human-genome-project

[28] http://geneontology.org

ent models can be embedded and re-categorized to acquire greater reciprocal transparency.

B.3.2 Types of Ontologies

Ontology 3.0 has two main classes: domain ontologies, *Ontology 3.0.1*, and domain-independent ontologies, *Ontology 3.0.2*. Domain ontologies are systems in that the basic entities and interrelations of a specific domain are introduced in machine-interpretable format by domain experts in order for a specific purpose held in the machine. They are divided into two groups according to the purpose of their establishment.[29] Application ontologies, *Ontology 3.0.1.1*, are software artifacts created for specialized information management purposes such as data storage, sharing, information extraction, and information representation (Cf. Jansen, 2009, p. 171). As-built for particular applications in the machine, the designs of application ontologies prioritize technical concerns, such as efficiency, ease of use, storage space. An example of this kind of domain ontology is the Bibliographic Reference Ontology,[30] formally defines bibliographic records and bibliographic references, and relates these into bibliographic collections, and relates collections of such records and references into ordered bibliographic lists (Peroni, Shotto, & other Collaborators, 2018). Reference ontologies, *Ontology 3.0.1.2*, on the other hand, are domain ontologies that contain knowledge representation of the most up-to-date studies in a specific domain. The primary purpose of constructing a reference ontology is to be reused in application ontology designs to solve incompatibilities caused by defining and/or naming the entities and relations differently. Thereby formulated in canonical syntax, definitions can be shared and used by different information system communities to support computational tools (B. Smith, 2014, p. 80). Thus, called a foundation ontology, a reference ontology is taken as analogous to a scientific theory (Jansen, 2009, p. 171). An example of a reference ontology can be the Anatomi-

[29] In the literature, there is a consensus neither on the kinds of ontology nor on the kinds of domain ontology. In this work, the classifications are designed according to our perspective.

[30] The notion 'reference' in its name may be misleading. That is an application ontology for bibliographic references.

cal Entity Ontology, AEO, which is an expansion of the Common Anatomy Reference Ontology (OBO Technical WG, 2018). AEO is an ontology of anatomical structures constructed for "facilitating annotation and enabling interoperability across anatomy ontologies" (ibid). Thus, AEO provides the basis for developing application ontologies, such as a research group can utilize AEO to support inference capabilities of its computational tools in clinical practices.

Designing robust domain ontologies as the software requires three interrelated fundamental characteristics: reusability, shareability, and interoperability. They maximize the artifacts' usability and provide representations of the reality that will depict the world in more detail day by day. In the end, the machine can process knowledge on a giant knowledgebase. To sustain these characteristics, designers should ponder on the following issues.

Domain-knowledge should be represented in such a way that it can be reusable. Suppose that there is an application ontology built for making inferences on appropriations and expenditures in May 2020. When it is established solely for May 2020, it reflects a closed domain, which cannot be reused for, say, May 2022. Ontologists must design their ontologies with such awareness. Related to this, ontology designs should be convenient to be pruned and enlarged. Suppose that an ontology of Turkish Folk Music is created in order to build a knowledgebase. It should be designed in such a way that one can reuse it for a high school-level inference system, and another can detail it to conservatoire-level knowledge representation. Each ontology speaks of scales employed in the music, yet in different levels of technicality. Thus, how we define the entities, relations, and properties and which formal language we use to formalize the axioms are of great importance in reusability.

When an ontology is utilized, sometimes it is necessary to add some new entities and/or relations, and accordingly new formal descriptions. Most of the time, it is solved case-by-case. On the other hand, when two different ontologies are supposed to be merged, each entity and relation cannot be revised one by one. This issue can be handled by creating bridge ontologies. However, it is costly since there may be a need for a new bridge ontology for each merging. In or-

der to sustain interoperability, controlled vocabulary is not always enough. A good solution comes from reference ontologies thanks to representing the upper categories in a domain and using the same formal language for representations. Created for different purposes, different ontologies can easily be merged when built upon the same reference ontology. A reference anatomy ontology can be used for bone-diseases ontology and tooth-decay ontology. Researches easily facilitate interoperability between these two application ontologies since the categories refer to the same things, and they are expressed in the same language. Nevertheless, what if one would like to work on two different reference ontologies? In other words, how are we supposed to sustain interoperability between reference ontologies?

Creating an ontology is toilsome. Deciding on the entities, relations, and properties, writing them in a formal system, defining axioms, and alike require intensive human labor. Ontology 3.0 cannot offer an automatized system for these works; therefore, shareability has upmost importance. For this reason, thanks to their being a *lingua franca* in a domain, ontologies are used to solve jingle-jangle fallacies that are very common in scientific studies. Jingle fallacy occurs when entities with identical IRIs/tags refer to different real-world phenomena; jangle fallacy occurs when different IRIs/tags refer to the same real-world phenomenon. On the other hand, an entity or a relation can be represented in different domains from different perspectives; thus, their ontological status or position in the hierarchy changes. That is an obstacle to share ontologies. Suppose that we want to build a graduate school ontology. We know that one graduate school calls "Ph.D. Thesis" and another calls "Dissertation" to the same entity. Since ontologies maintain a thesaurus, two different labeled entities are known to be referred to as the same entity. What if, in their formal definitions, the entity "Dissertation" contains temporal aspects, whereas "Ph.D. Thesis" refers to a bunch of printed papers? How should ontologists think of the philontological status of the entities and formal properties of relations?

The questions in the last two paragraphs paved the way for building domain-independent ontologies. The second class of ontologies, *Ontology 3.0.2*, aims to establish semantic interoperability among reference and application ontologies

by introducing common categories across all domains. For the focus on the highest categories of being, these ontologies are also called upper-level ontologies (ULOs). ULOs define and axiomatize the most general categories, such as objects, properties, relations, events, space, time, and alike, to serve as an underlying structure of any domain ontology at any level any granularity. Moreover, modes of being, such as space-time, part-whole, and border, are also defined and axiomatized in ULOs. For this purpose, they are to be constructed with rich axiomatic systems for establishing the meanings of all kinds of categories. For instance, in temporal logic, it is a matter of choice of taking time as time points or as time intervals. Thus, the category of time should be established as such axioms in ontologies both in robotics and in quantum mechanics can give the meaning of the reality in both fields with the same formal language. The most popular upper-level ontologies are the Basic Formal Ontology (BFO), the Suggested Upper Merged Ontology (SUMO), and the Descriptive Ontology for Linguistics and Cognitive Engineering (DOLCE).

Upper-level ontologies are a framework for reference and application ontologies. [31] For instance, the above-mentioned Anatomical Entity Ontology is one of the Open Biological and Biomedical Ontology (OBO) Foundry ontologies that are built upon the Basic Formal Ontology (BFO), one of the most used ULOs. Developed collaboratively, the OBO Foundry consists of domain ontologies, ontologies that computationally represent our biological and biomedical knowledge in various levels, granularity, and perspectives (OBO Technical WG, 2016). Building ontologies with an upper-level ontology guarantees the three fundamental characteristics of Ontology 3.0, interoperability, shareability, and reusability. Furthermore, the most arduous deed of constructing the family of reference or application ontologies in a logically well-formed and scientifically accurate way is warranted by the selected ULO as well. However, in general, ULOs provide basic ontological restrictions through axioms so that a ULO can

[31] Quinean ontologies resemble reference ontologies, for they reflect the domain knowledge with formal means. Whereas Husserlian formal ontologies resemble upper-level ontologies as the top-level categories and relations are defined formally, the construction becomes applicable to any domain. Another interpretation is that Quinean ontologies can be regarded as Husserlian regional ontologies.

also be used as a means to verify the plausibility of and compatibility within a set of domain ontologies. Consequently, a ULO establishes a foundation upon which semantically interoperable ontologies can be developed collaboratively.

Lastly, we want to highlight the indispensable aspect of philontologies in building reference and upper-level ontologies. Reference ontologies must reflect the highest categories of a domain and express the latest knowledge in a formal structure. Upper-level ontologies must reflect the highest categories of being and express them in a formal structure. Thus, skills in representing knowledge of the phenomena are essential in designing ontologies. That is, appropriate philosophical theories must take place in the construction of reference and upper-level ontologies. Yet, the use of these theories in building an ontology is also shaped by the requirement that ontologies must be consistent with scientific facts and that these ontologies are created to be used in information systems. Regardless of building an application or a reference ontology, defining classes and assigning properties requires a philontological solid perspective. Representation of colors, for instance, can be regarded as 'quality' or 'abstract,' where quality can be taken as dependent on a concrete, whereas abstract can be taken as ontologically independent from concrete. Which ontological stance for universals is taken is of crucial importance. Further, along with philosophical concerns, an ontologist should be able to design their ontology by being aware of the formal languages' limitations. A formal language is expressible enough for mereological facts should be a concern for an ontologist. In the following part, we will explain this requirement in detail with implementation procedures of ontologies.

B.3.3 Building Ontologies in Ontology 3.0

There are various ways of representing knowledge in the machine. However, ontologies are the most robust ones for they not only provide standardization in a domain but also inference applications. In this section, we are going to examine how to represent knowledge with ontologies. Yet, let us begin with a caveat. In truth, there is no particular or correct methodology for developing ontologies in general. Domain ontologies are primarily designed with considerations of clients' requests, computational restrictions, and/or specifications of application

domains. Hence, domain ontologists keep in mind the prerequisites, requests, and restrictions of application-users, -domains, and -clients while developing their ontologies.[32] In other saying, a domain can be represented from different perspectives. Or, it can be modeled differently, even for the same purpose due to the ontologists' different ontological views.

Design of an ontology starts with declaring the reason for building the ontology at hand. Stating the reason can be thought of as laying a foundation of a building. As the whole building is going to be constructed on such a foundation, the ontology will be built upon its construction purpose. Representing knowledge of a bank's ATM transaction processes to optimize all the bank's transaction processes is an example of an application ontology. The ontologists mainly focus on the entities of ATMs and transaction processes particular to the bank. All the relations and properties of the entities are selected relative to transaction processes taken on the bank. Suppose that the bank managers realized the need for various application ontologies to meliorate the bank's operations. For these applications, ontologies interoperate easily; there should be a reference ontology. If the reference ontology is asked to be domain-specific, then, again, the entities, relations, and properties are selected concerning operations held in the bank. The entities, on the other hand, must refer to the highest categories of operations. As a second option, an upper-level ontology can be used as a framework for application ontologies. Since a ULO is domain-independent, how can a ULO be chosen that is appropriate for the bank? Laying a foundation of an ontology is also necessary for ULOs as well. For instance, whether abstract entities and/or universals are included in the ontology is up to the philosophical stance of the upper-level ontology designers. If the designers admit fictional entities in their ontologies, they must include a category for them, or they do not admit abstract entities in their ULO; they need to guide practitioners on representing numbers in the applications. Hence, if a ULO is designed to overarch all the

[32] As Noy and McGuinness (2001, p. 2) point out that designing ontologies and object-oriented programming seem similar, nevertheless their perspectives differ. In an ontology design, what is essential is *structural* properties of a class, which consists of entities and relations, where, as in object-oriented programming, what is designed is *operational* properties of a class, namely the methods of a class.

entities in the scientific domains, the designers select the highest categories in which scientific entities fall. Suppose the designers of a ULO are philosophically realists, in the sense that the general features of reality are known in the form of universals and the relations between them. In that case, selecting the highest categories of being and their formal representations must be in accordance with the universal-particular dichotomy. Finally, a ULO should be selected that is convenient to entities in the applications of a bank. For instance, BFO would be a better choice than DOLCE since BFO explicitly accounts for temporality, whereas DOLCE does not allow temporal indexing, which is essential for transactions.

As mentioned in the last paragraph, declaring the purpose for building an ontology simultaneously and implicitly reveals which entities should be selected for building the ontology. Then, the second step of building an ontology is selecting entities, their interrelations, and their properties as relevant to the purpose for building it. The most common general terms taken from several sources, such as books, articles, terminologies, and relevant ontologies, are listed, then selected in accordance with the subject matter of the ontology. The third step is defining and organizing all these materials: constructing a taxonomical hierarchy, exhibiting semantic webs, and assigning the properties. Taxonomy is the backbone of the ontology since it gives all the subsumption relations and provides a base for exhibiting semantic webs. Ordering entities in a taxonomy identifies the highest categories and ensures coherence between them. Moreover, it facilitates defining the entities that provide ease for human readability and for writing axioms. The next step is the implementation of the structure into the machine.

Everything represented in the machine has a unique identifier.[33] Just as unique identifiers of artifacts in libraries let us find a book at its proper place, IRIs let the machine trace entities and their related entities. Entities with IRIs in the taxonomy and the semantic web are encoded with tags and annotations, which

[33] Refer to section From 1.0 to 3.0: Industry, Science, Web

are the most important for sustaining machine-readability. Ontologists need to decide on a formal language to encode their ontologies. There are various formal languages for representing semantics in the machine, the Web Ontology Language (OWL), the Resource Description Framework Schema (RDFS), Knowledge Interchange Format (KIF), Ontolingua, and DAML+OIL, to name some. Which formal language to be used hinges on the purpose of the construction of the ontology. Designers inspect the expressibility and the complexity of the formal language: the less complexity and the more expressibility are inherited in the optimum formal languages. Further, selection of the formal language also determines which ontology editors[34] and semantic reasoners[35] to be used. Ontology editors are used for encoding the ontology; namely, entities, relations, properties, and axioms of the ontology are represented in the editor. Semantic reasoners inherit the inference rules and are embedded into ontology editors. They operate on the axioms and declarations to check the structure's consistency and/or make inferences. Then, once the ontology is represented consistently via an ontology editor, the artifact is ready to be used following its purpose for construction.

[34] Protégé, OBOEdit, OntoEdit, Fluent Editor, NeOn Toolkit, OWLGrEd are examples of ontology editors.

[35] The reasoning programs that in standard use with OWL are Pellet, FaCT++, and HermiT OWL Reasoner (Arp, Smith, & Spear, 2015, p. 174).

C. CATEGORIES THAT DEPICT THE WORLD

One of the aims of this work is to depict the world in the machine. To this end, one of the first steps should be understanding how we depict the world for ourselves. The purpose of this appendix is to give a philosophical background for an examination of the phenomena. Thus, this part pictures what beings and their interrelations are and the integrity and diversity of meanings through constructing philontologies. In this sense, a philontology is a means of labeling something as being and listing the semantic properties of those that exist. That said, this part also shows our philontological investigation through which we came up with the idea of employing process of semantic properties into computation.

What determines the structure of an ontology is its categories. Aristotle states his categories by asking questions like "how much," "where," or "in what condition;" in contrast, Quine constructs his ontology on a maxim that "to be is to be the value of a quantified variable." The first part of this appendix will investigate the categories that help us depict the world. Through analyzing these categories, in the second part, we will decide on which category or categories would be useful for representing integrity and diversity of meanings. Again, the fundamental structure that reveals semantic properties is ontology, and ontologies are constructed on categories.

Please, consider the following example. When an entity is under the superclass of the abstract, whatever an abstract means, the semantic properties of the abstract are ascribed to the entity, and all the possible relations that an abstract entity can have are determined inevitably. Suppose that abstracts are defined as intangible entities which cannot be twisted or heated. If numbers are abstract, twisting or heating cannot be in the collection of semantic properties of numbers.

Again, the world is depicted via ontologies, whose essential elements are being; however, the aim of this appendix is not to construct an ontological system that transforms phenomena into data. On the other hand, it aims to pave the way for a depiction of the world that contains (1) a collection of semantic properties of an entity and (2) the possible operations among semantic properties. To sum up, in this appendix, we will investigate how we understand the world, how we classify the entities that exist, and the differences between them. In the end, we will have learned the philosophical position of the trope theory.

C.1 Abstract-Concrete Dichotomy

Defining the abstract-concrete distinction, if any, is one of the main problems in philosophy. Accepting or rejecting the existence of such entities necessitates thorough definitions, through which we can get an insight into how entities get semantic properties, "abstract" and "concrete."

As concrete is trivial, the question of the principles of representing an entity as abstract is the first to be answered. Williams (1953, p. 14) confronts us with the ambiguous and various uses of the term. An abstract object can be either "the product of some magical feat of mind" or "the denizen of some remote immaterial eternity," or,

> The abstract is equated with the abstruse, the ethereal, the mental, the rational, the incorporeal, the ideally perfect, the non-temporal, the primordial or ultimate, the purely theoretical, the precariously speculative and visionary; or again with the empty, the deficient, the non-actual or merely potential, the downright imaginary, and the unreal. [...] Mathematics or logic is called "abstract" partly because it is about formal structures, partly because it treats them only hypothetically; but a symbolic calculus is called "abstract" because it isn't about anything.

The most common definition is that abstracts are non-spatiotemporal (Rodriguez-Pereyra, 2019): abstract entities exist nowhere in a specific time. Accordingly, concrete entities have both spatial and temporal extensions. For example, mathematical objects, e.g., a set, and linguistic concepts, e.g., a concept, are examples of abstract entities. Falguera, Martínez-Vidal, and Rosen (2022) states that chess must resemble a mathematical function in its relation to space and time. On the other hand, (Rodriguez-Pereyra, 2019) considers games non-spatial but

temporal. We have two options: either we accept that abstract entities distinctively exist in space and time as concrete entities do (Falguera et al., 2022) or reject the entities with temporal aspects as abstracts. The first option leaves us with the problem of defining the "distinctive ways." Think of a set of flowers in a specific room. It has 7 elements for the time being, yet some of the flowers will cease to exist after some years, and there will occur a new set with the element number, say 5. Alternatively, please think of the space that this flower set occupies: it is untenable that it occupies as much as space that the flowers seize. However, this set of flowers cannot occupy a location and exist at a particular time. Sets as pure mathematical entities cannot exist in space and time, whereas sets consisting of concrete entities –the impure sets- can be spoken of their location. Here, we need to choose between two options: either we accept that impure sets as abstracts exist in space or reject abstract entities cannot occupy space since impure sets are abstract (Falguera et al., 2022). Either way, there is puzzlement: electrons do not occupy a region in space, according to some quantum scientists (Falguera et al., 2022). Are electrons abstract or concrete?

This question compels us to examine another definition of abstract: abstracts are causally inert entities since only spatiotemporal entities have causal relations (Rodriguez-Pereyra, 2019). If electrons are abstract, they do not have causal powers, which is untenable; if they are concrete, concrete entities do not occupy a region in space, which is a contradiction. Leaving perplexity of quantum entities aside, let us discuss which entities can be regarded as causally inefficacious. The above definition states that only concrete entities have causal powers. Consider the event of Brutus' stabbing Caesar. This is an event and can be considered concrete since it happened in a particular place and a particular time. While the causation of this event is non-spatiotemporal, thus abstract (Schaffer, 2016). To solve this problem, we can group causation as concrete and abstract only in terms of causal inefficacy. That means there can be concrete entities, whether mental or material and only those have causal powers. Then, abstracts are numbers, sets, and alike that make nothing happen. Both pure and impure sets are abstracts at that rate since they make no definite causal influence on what emerges (Falguera et al., 2022).

Uzun İhsan Efendi, the protagonist in *Puslu Kıtalar Atlası*, is a product of İhsan Oktay Anar's mental activity. As a mental entity, does it have any causal powers? If any, what are they? Thus, is Uzun İhsan Efendi an abstract entity? In the book, Uzun İhsan Efendi surveys truth in his dreams; in other words, he uses dreaming of searching for truth. We can claim that the idea that truth can be surveyed in dreams may trigger someone to practice dreaming as Uzun İhsan Efendi did. Can we conclude that the character of a book can be a reason for this practice? We affirm without hesitation since events are concrete; either they have spatiotemporal aspects or are causally efficacious. When we accept it as a concrete entity, it must trigger anyone who reads to book in the same way. No prima persona affects two people in the same way. Thus, some causal powers have multiple effects simultaneously. As a consequence, that abstract entities have causal powers must be tenable.

Besides, electrons are concrete since they have causal powers. Obviously, they are not mental since they are mind-independent entities. So, they are concrete. We need to revise the definition of abstract once more: mind-independent entities are abstract. When we strip being abstract from causality, is being mind-dependent sufficient reason for being abstract? Or, can being concrete be equated with being mind-independent? Physical entities, for sure, are mind-independent ones. On the other hand, Platonists claim that mathematical entities are mind-independent, which makes them concrete. However, this is never true since mathematical objects, like other Forms, are intangible. Hence, there should be another criterion to differentiate abstracts from concretes: tangibility. Let us attempt to define abstract entities: an entity is abstract if and only if it is mind-dependent and intangible. In this sense, mathematical objects can be concrete due to their being mind-independent,[1] or electrons or quarks can be abstract due to their being intangible [?]. The double ontological state –from one aspect concrete from the other abstract- is unacceptable.

[1] On the other hand, sets are mind-dependent when constructing them.

The problems occur due to distinct ontological stances. Being concrete, which necessitates having causal powers, is almost always equated with being tangible. What is tangible is always in space and time. From the other way around, what is not in space and time is intangible; thus, an abstract. We must deny the existence of impure sets, quarks, and even the dreams that exist in time yet intangible. Or, we need to choose a side by either accepting the mind-independent existence of mathematical objects or claiming that mathematical objects are mind-dependent. Nevertheless, we want to offer the oldest philosophical way to differentiate abstract and concrete: abstraction.

Here we can use Aristotle's approach: the beings that are separable in thought are abstracts; the beings that are separable in reality are parts. According to this approach, both parts and abstracts necessitate another entity for their existence. Namely, an abstract is an entity that cannot exist separately from and that is taken as a separate entity by our abstraction faculty. When we look at a specific iron globe, we cannot separate its sphereness from the globe, yet sphereness can be separated from the globe in thought. The principle of separability in thought is not the only way to formulate abstraction. We can apply abstraction functions to concrete entities, and an abstract entity is *essentially* the value of the abstraction function (Falguera et al., 2022). Note that there cannot be a function that asks for arbitrary values; in other words, what is abstracted from a concrete entity is defined by the function. For instance, Color(this apple)=red; Capital(France)=Paris; Shape(Sun)= sphere; Quantity(Humans in Venus)=0. The value of the abstraction function *Color* must be a color; similarly, the value of the abstraction function *Shape* is essentially a shape, whereas Paris is not an *essential* value of the abstraction function *Capital*. As an abstraction function gives the *essential* values due to the abstraction criteria, the idea of applying the abstraction function is not helpful in all circumstances.

One may ask what we will get at the end of applying all the possible abstraction functions to an entity. If the concrete entity is an ontologically independent being that attires to abstracts, what is left when we stripe all the abstract entities? We will return these questions in the following parts, but let us consider whether we can formally model this metaphysical distinction.

We could not have figured out whether we should include spatial and/or temporal aspects into concern; similarly, it is doubtful whether we should consider including tangible and intangible distinctions. Let us use the ontological dependency criterion, which seems the most useful for formalization.

Rojek (2008) uses the ontological dependency criterion for distinguishing abstracts from concretes. He explains the relation between abstract and concrete with inherence. Inherence is a relation with a property of order; namely, it should be reflexive, transitive, and nonsymmetrical. It is legitimate to claim that if an entity, say x inheres in another entity, say y, then x is more concrete than y (Rojek, 2008, p. 362). For instance, redness is more concrete than color. Hence, the existence of an abstract necessitates a concrete entity for its existence. When ontological dependency is formalized with inherence relation($\leftarrow$), an absolute abstract entity, x can be defined as "$\exists y(y \leftarrow x \wedge x \neq y) \wedge \forall z((x \leftarrow z) \Longrightarrow (x = z)))$"; where x is an absolute abstract entity such that there is a distinct entity inheres x and for all other entities that are inherited in x requires that x is the same entity with the other entities. Rojek's further analyses suggest that we can speak of an absolute concrete entity with the same formula. Moreover, these formulae also suggest that an abstract entity can be concrete simultaneously, so being abstract does not necessarily exclude being concrete. For sure, there can be different formal modeling of abstract-concrete dichotomy. The point here is that the ontological dependency of abstracts cannot be formalized without regard to the levels of concreteness. For example, a cup handle, which is concrete for sure, is ontologically dependent on the cup, yet a cup can exist without a handle. On the other hand, a handle does not inhere in a cup; a handle can be a part of a cup. Then, how can we differentiate inherence relation from parthood relation? A thorough mereological investigation is needed. This proposes that the abstract-concrete dichotomy is relative. Thus, in which circumstances could we apply abstractness or concreteness to an entity as a semantic property? Before answering this, let us dig into what Rojek (2008)'s absolute concrete entity can be.

C.2 Substance Theory

When the abstract entities are ontologically dependent on the concrete entities, what is left when we stripe all the abstract entities from an entity? There can be many philosophical answers to it. One may reject the idea of separable in thought, or one may claim that nothing is left, or one says that there is something left, which is "something that stands under or grounds things" (Robinson, 2021). According to the last approach, the thing that is the foundation of the concrete entities is called *substratum*. Kantian way of describing substrata is defining them as *Dinge an sich*, or *noumena*, observation-independent entities. Recall that these metaphysical entities are unknown to us.

Let us remind the reason for our metaphysical investigation: we are trying to find an ontological system that helps us depict the world into the machine. It seems that there is no apparent motive for accommodating substratum either for us or for the machine since we cannot represent something that we do not know. On the other hand, in philosophical parlance, the term substratum can be used in its most specific sense: an individual. So, it is better not to rule it out at once. In the narrower sense of substratum, it is usually translated into English as "substance."[2] This kind of substance, then, is not the answer to the question of what is left when all the abstracts are stripped from a concrete entity. Rather, this is another way of speaking of the concrete. Thus, in philosophy sketchily, there are three main substance theories. Firstly monists, for who there is only one substance in the world within which all being is infused. Pneuma of stoics is an example of monistic substance theory. Secondly, dualists claim that there are two distinct fundamental substances: material and immaterial substances. These two distinct substances have different identity conditions, which form an entity (Robinson, 2020). The third substance theory affirms a good number of substances. Aristotle's hylomorphic categories are the best exemplary.

[2] The term substance is also used as an English translation of *subsratum*, in the sense of observation-independent entity. Be aware of this ontological distinction.

Which substance theory -monist, dualist, or pluralist- fits our ontological investigation? We would not eliminate the monist approach for depicting the world in the machine, for the approach seems quite metaphysical, if not mystical, burdensome for our intellect. The monist approach will be investigated at last because we need to figure out all the possible categories used for picturing the world. Dualism, on the other hand, does not necessitate such a suspension. In the debate of abstract entities, we encountered that describing mind-independent entities is not philosophically obvious. The metaphysical stances characterize the features of mind-dependency or mind–independency. A typical example is the ontological status of mathematical objects.[3] The last approach, pluralist substance theory, admits multiple ontological categories. In Aristotelian ontology, on the one hand, there are universals vs. individuals/particulars as one categorical dichotomy, and on the other hand, there is substantial vs. accidental dichotomy. This schema is called Aristotelian ontological square. Before digging into this square and other polycategorical ontologies, we need to investigate the nature of universals and particulars.

C.2.1 Universals versus Particulars

In philosophy, it is pretty easy to talk about particulars: as far as we can point at an entity that is particular. "This man," "this opinion," "these colors" are examples of particulars. Universals, on the other hand, are the entities that can be instantiated either by another universal or a particular; i.e., universals are the only entities that can be instantiated. Therefore, the demarcation relation of universal-particular distinction is instantiation.[4] The existence of universals is one of the oldest debates in philosophy. Realist admits universals exist: Ante rem realists maintain universals as non-spatiotemporal entities, namely univer-

[3] In Web 3.0, some ontologies omit the mathematical objects or are constructed without philosophical concerns. In Ontology 3.0, ontologies either inherent mathematical objects in various categories or reject their existence. However, we cannot rule out such objects since, *one way or another*, we have to represent them in the.

[4] A confession is on the way. In this appendix, we speak of categories regardless of clinging to any philosophical doctrine. We never choose a philosophical approach and follow its categorical structure to understand and picture the world. We are rather searching an appropriate way to represent entities in terms of their semantic properties, while considering other ontological categories that are related with properties.

sals exist independent of their instances; whereas in re realists, maintain the existence of universals in space and time. One sect of the nominalists rejects the existence of universals (Rodriguez-Pereyra, 2019).[5] The last camp of the debate is conceptualists, who deny that mind-independent universals exist. They rather label universals as "non-linguistic mind-dependent entities," namely as concepts (Orilia & Paolini Paoletti, 2020). When the existence of universals is affirmed, how are we supposed to interpret such entities: abstracts or concretes, or something distinct from these two? Rojek (2008, pp. 365–366) speaks of three concepts of universals. The first one is universals as common properties. The above-stated definition of universals acknowledges that universals are instantiated entities. One interpretation, then, can be universals inhere in many universals or particulars. Taste inheres in both sweet and sour; sweet inheres in both baklava and cannoli.[6] The second one is universals as indeterminate entities. Particulars and/or properties make these indeterminables determinate. Particular sweetness and taste are fundamental, and such particular sweetnesses determine universal sweetness. The last one is universals as inhered entities. Universal sweetness is inhered in this piece of baklava and this cannoli.

In sum, there seem two relations for universals exist: inherence and determination. Nevertheless, inherence relation has two distinct senses in the first and the third depiction of universals: is universal to be taken as a domain value or co-domain value? Moreover, along with the definitions, we also need to consider the formal structure of the categories. Thus, how are we supposed to determine the relation type of the universals?

Supporting a solid ground for universals is troublous; it is so for particulars. When we maintain that universals exist in space and time, they have attributed concreteness. Again, when we admit that abstracts are non-spatiotemporal entities, is the sweetness of a specific piece of baklava universal or particular? Thus, the sweetness of baklava is universal property, just as sweetness and the sweetness of this piece of baklava is a particular property. Further, we should not state

[5] The other rejects the existence of abstract objects. Cf. Falguera et al. (2022)

[6] Sweet as a semantic property is in the collection of semantic properties of baklava and cannoli.

something like universal substratum and particular substratum if we accept such an approach. Furthermore, Robinson (2021) notes that most philosophers, if not all, accept using substance concepts in their specific usage since individuals are essential for us to understand the world. Substance concepts roughly are what we generalize from the individuals, such as *human* or *computer*.[7] Do we need to acknowledge substance concepts as abstract or as universal? There is no plausible answer since it is still unclear how we define abstract-concrete entities or universal-particular entities.

C.3 Properties

When universals are held as common properties, what are the "uncommon" properties? Furthermore, what is a property? Are properties mind-independent or ontologically dependent? Are they abstract, so are they universals? The perplexing realm of metaphysics pulls us in again.

Properties, another ontological category, are entities that entities exemplify or instantiate (Orilia & Paolini Paoletti, 2020). Socrates instantiates the property of being human, and being human exemplifies the property of being a living being. The sweetness of this cannoli exemplifies the property of being sweet. Obviously, there are various kinds of properties. Taken from Orilia and Paolini Paoletti (2017), some of them are listed below.

- *Primary and secondary properties*: Primary properties are mind-independent entities, such as shape, size, or charge, whereas secondary properties are mind-dependent entities, such as smell, color, or taste.

- *First-order and higher-order properties*: First-order properties can only be instantiated by individuals, such as baklava is sweet or the dress is crimson. On the other hand, higher-order properties can be instantiated by first-order properties, such as sweet exemplifies the property of being

[7] They are often called *sortals*.

taste; crimson exemplifies the property of being red and the property of being a color.

- *Typed vs. untyped properties*: Following Russell's type theory, typed properties are precisely partitioned into levels, such that there is a well-formed hierarchy in properties. However, to accommodate self-exemplification and/or transcendental properties, such as "thinking about," typed theory should be replaced favoring untyped or type-free property theory.

- *Simple and compound properties*: Properties either are only simple or are simple and compound. The former takes all the properties as simple; the latter accepts that simple or simpler properties can form other properties. Understanding all of the properties as simple is straightforward; however, explaining how the simple or simpler properties form compound properties requires a formal analysis. This brings another kind of property:

  - *Structured vs. unstructured properties*: Some properties are structured by logical or some other formal connectives.

- *Determinates and determinables*: Properties are distinguished from each other by determination relation, which gives the first instances of type-level properties. For instance, *sweet* is a determinate of *taste*, or *red* is a determinable of *crimson*.

Apart from these kinds, there are other kinds of properties as well: propositions, tensed properties, sortal vs. non-sortal properties, genus and species, natural kinds, purely qualitative properties, essential properties and internal relations, intrinsic vs. extrinsic properties, supervenient and emergent properties, linguistic types, and categorical properties vs. causal powers (Orilia & Paolini Paoletti, 2017).

Philontologists admit some of them and reject the rest. There cannot be an ontology that maintains both determinates and first-order properties since the first-order properties are instantiated by individuals. In contrast, determinates and determinables cannot be assigned by the instantiation relation but rather by the determination relation. Only higher-order properties can be determinate

or determinable properties, yet the philontologist must prove that the ranges of determination and instantiation functions are one and the same. For instance, the property of being taste is a higher property since another property can instantiate it; and it is also a determinable since its more specific versions can be found. Then, it is instantiated and determined by the same properties, such as the properties of being sweet, sour, or spicy. A lexical analysis offers that a determinate is ontologically prior to a determinable: an agent must be ontologically and logically prior to an act. That follows; the property of being sweet is ontologically prior to the property of being taste. Is it legitimate to transfer this ontological status to first and higher-order properties? Just as a realist can claim, there can be no such priority relation between them. The bottom line is that a philontologist must consider all of the categorical aspects when building an ontology.

Orilia and Paolini Paoletti (2017) speak of relations as a kind of property in the sense that relations can be reduced to monadic properties. Commonsensical differentiation between properties and relations is that the former holds *of* the particulars, while the latter holds *between* the particulars (Cf. MacBride, 2020). However, argument deletion lets us reduce dyadic relations into monadic ones. For instance, "Laura is eating a piece of baklava" offers that "is eating" relation is dyadic; whereas as argument deletion transforms this sentence to "Laura is eating," in which "is eating" relation is a monadic property.[8] Hence, even Laura has the relational property of bearing "eat" a piece of baklava, while baklava has the property of Laura's bearing "eat" to it. That is, even if "to eat" is binary, the relational property "is eating" is unary.

Moreover, predicates should be a part of our categorical investigation of properties and relations. G. F. Stout (1940, p. 117) defines a thing as a complex whole that includes all properties that truly predicate it. That is, a predicate is a constituent of a property complex (p. 118). In this sense, a truly predicable is a property of a thing. Further, relations play a crucial role in knowing a thing:

[8] Orilia and Paolini Paoletti (2017) highlight that "is eating" relation necessitates *something* for monadic property to be a relation: "Laura is eating something."

non-relational properties cannot exist without the relation to other things or properties (p. 120). Therefore, both relational and non-relational properties are parts of property complexes. As the connection between properties, predicates, and relations is crucial for our investigation, we will dedicate a separate section to investigate this connection in the following.

C.3.1 Several Senses of Properties

The history of philosophy is full of opposing ideas, so the theory of properties is not an exception. According to Aristotle, properties are separable in thought: redness as an entity is only separable in thought and cannot exists without its bearer. Thus, properties are both abstract and ontologically dependent. Moreover, Locke elaborates this and specifies a category as secondary properties, which are mind-dependent entities. That makes some properties ontologically prior to others. Plato rejects such a view and maintains properties, or universals, exist separately and are capable of being instantiated by different entities. However, Williams disagrees with the multi-location of properties and admits all the properties are particulars. Moreover, in the literature, some categories are equated with properties, such as universals, abstracts, and predicates. Besides, some categories are used in the sense of properties, such as Locke uses the term quality and Stout uses the term character in the same sense with the term property (Falguera et al., 2022; G. F. Stout, 1940). Examining the category of properties also requires examining relations between the mentioned categories, such as the relation between universals and properties, character and properties. This is crucial for us to define all the fabrics of the world so that we can manufacture the world into the machine with properties.

The terms property and attribute are used interchangeably. On the other hand, a linguistic analysis offers that an attribute is predicable, and the term "property" points to ownership. In other words, a property that is attributed to an entity is owned by or is in the entity itself. In this case, a property is an attribute of an entity (Falguera et al., 2022), which makes an attribute is more than a property. Moreover, Rojek (2008, p. 370) defines another term, "aspect," with which

properties constitute the set of attributes. He provides etymological analysis of the term "aspect": The term is connected with "seeing," which requires an agent who constructs the attribute by her cognitive faculties. The aspect-property dichotomy is equated with the determinable-determinate dichotomy.[9] However, Levinson (1980, p. 102) uses the quality-property dichotomy instead of the aspect-property one. Both Rojek (2008) and Levinson (1980) mention both as generically attributes.

When properties are taken as abstract entities, then every property exemplifies abstractness (Orilia & Paolini Paoletti, 2020). Besides, properties can, for sure, exemplify properties, such as the property of sweetness exemplifies the property of taste. Moreover, being abstract is a property, so it exemplifies itself. Although this approach is legitimate to many metaphysicians, Russell rejects the self-exemplification of properties because such exemplifications cause paradoxes in the formal representations.

When properties are taken as universals, how are we supposed to explain the one-over-many problem of universals? For example, colors of two different concrete entities, say apples, can be distinguished: one is red, the other is green. What can we say about two green apples of equal size? These apples are numerically different, yet they are somehow one. How can the properties of size and color be in many places? One may state that there is nothing wrong with equalizing universals and properties, and we can solve the one-over-many problem as we solved it for universals. We need to wait until the decision of which categories appeal to be crucial for a machine ontology. We can skip this question for the time being.

Inevitable questions bear in the minds of ontologists. How can we differentiate these categories used interchangeably, and how can we formalize nuances of these categories? What Locke calls quality is different than what Levinson calls it. Further, when simple properties are unanalyzable, we cannot talk about

[9] The difference, however, lies in the demarcation function used. For the former inherence relation differentiate aspect and property: properties are inhered in entities, whereas aspects are not; for the latter determination function is used, as stated above.

an inner structure of them; yet complex properties are analyzable so that they have an inner structure. What is the methodology of property analysis? How can one ensure that this method will not be circular or linearly ad infinitum? If there is an inner structure of complex properties, can such a structure be formalized? When the philosophers are so lost in defining these categories, how are we supposed to represent them in the machine? No matter which philosophical aspect we entertain, such an abundance of categories, which will grow more and more in the following pages, increases the computational load as well. In a nutshell, we are tussling with metaphysical abstrusities once again.

C.4 Relations

As a category, relations are included in many philontologies. For instance, Aristotelian realism embodies relations in some categories such as relative, posture, state, and passion; Kant speaks of relation as a category by itself; Husserl describes relations under the ontological categories correlated with semantic categories; updating Aristotelian categories, Grossman calls relations one of the eight highest categories, in which he distinguishes properties from relations. Further, Hoffman and Rosenkrantz take relations as abstract entities, along with disjoint entities of properties and propositions, and Lowe distinguishes particular relations from universal relations (Thomasson, 2019). A realist can accept properties as universals; yet can reject the existence of relations. Accordingly, for a nominalist, relations are out of the scene. If so, how come can two opposing ontological doctrines share a common conclusion about relations? Are relations abstract? Are they particular? Are they distinct from properties? Are they universals? How can they occur in various categories? Could we speak of a representation without relations? In what follows, we will investigate relations qua an ontological category.

Properties are defined above as entities that are exemplified or instantiated by some other entities. As noted before, they are monadic since properties are only exemplified by other entities, either by particulars or by properties, on an individual basis. In comparison, relations are considered n-adic, where $n > 1$, since they are exemplified between particulars. In logic lectures, properties are

formalized as $F(x)$, while a dyadic relation is formalized as $R(x, y)$, where F for a property, x and y for arguments, and R for a relation. What if $x = y$, is $R(x, x)$ a relation or a property? MacBride (2020) highlights that identity is a relation, not a property, so the terminus of the relation can be the same as the subject. The main distinction, thus, lies that relations, on the contrary to properties, hold *between* things, even in some cases it connects the entity at hand with itself.

There is a perplexing entity called "relational property." A *relational property* is a property that depends on a relationship between entities. That is, it is "a predicate of a given entity which is generated in it by the subsistence of a relation between that entity and other entities or itself" (Findlay, 1936, p. 176). Recall that identity is a relation. At the same time, there is a relational property of identity in the entity to its terminus. Findlay (1936, p. 176) explains the difference with an analogy: the "relations are bridges between entities, whereas relational properties are the points of contact between the entities and such bridges." Thus, identity is dyadic, but the relational property of identity is monadic. To make it more concrete, consider the following examples. Marriage is a relation between two humans, but 'being married to someone' is a relational property. If a book is on a table, there is a certain spatial relation between the book and the table. This spatial relation generates in the book the relational property of being on the table, and in the table the relational property of being under the book.

C.4.1 Internal and External Relations

Relations can be distinguished as internal and external. An internal relation is a relation whose holding between entities is somehow determined by the entities; an external relation is a relation whose holding between entities is not determined in a specific sense. Philosophers disagree on how internal relations are determined. As Moore admits, an internal relation is determined by the existence of the entities that it relates; as Armstrong claims, an internal relation is determined by intrinsic natures, i.e., non-relational properties, of the relata; as Lewis states, a relation is internal if it supervenes upon the intrinsic natures

of the entities it relates (MacBride, 2020). Beyond any doubt, what an external relation is a matter of debate as well.

There is eliminativism about external relations and reductionism about internal relations, both of which are controversial. Bradley's Regress, the most famous eliminativism about external relations, concludes that we should exclude external relations from our ontology because each external relation necessitates another external relation to explaining the previous one.[10] This ad infinitum regress makes such relations unintelligible. In keeping with the definition of internal relation, recall that either the mere existence of the relata is necessitated or the relation must supervene upon some intrinsic characters of the relata. The first case offers nothing about the internal relations: Claiming their existence is nothing since, in Armstrong's parlance, it does not constitute "an addition to the world's furniture" (MacBride, 2020). The second case, on the other hand, seems promising, since matter and charge are the most common examples of intrinsic characters of any physical entity (Lewis, 1986, p. 14, p. 60). However, quantum theory acknowledges that we should eliminate intrinsic properties from out of metaphysical outlook (Teller, 1986 from MacBride (2020)).

C.4.2 Order and Direction

We will mention the nature of relations, viz., order and direction. Recall that a relation, R is symmetric if $aRb = bRa$; is non-symmetric if $aRb \neq bRa$. How can we decide whether a relation is symmetric or not? Is it up to the entities it relates to, or does a relation bestow order upon its entities? If the former is true, we need to figure out the natures, namely per se, maybe also per accident, attributes, of entities, and which attributes admit/allow/emerge what relations with respect to the other entity's attributes. Otherwise, for the latter, we search for the converse of the relation and check whether the relation and its converse are one and the same. Note that finding a converse of a binary relation is more or less easy, whereas can we speak of a/the converse of an n-adic relation? So,

[10] See F. H. Bradley ([1893] 2016) for the discussion in detail.

admitting an order of the entities in a relationship is both computationally and metaphysically important.

Let us focus more on how to differentiate symmetric relations from non-symmetric ones; in other words, how to determine the direction of the relations. It is legitimate to define symmetric relations as relations whose converse is itself. Suppose that R is a relation between a and b, and that $R*$ is the converse of R: if aRb, then $bR*a$. Assume that if R and $R*$ are one and the same, so as the mentioned two states. For instance, the book's being on the table is the same state as the table's being under the book; these two states are one and the same.

When we interchange the relations – the book is under the table, and the table is on the book – they express one and the same state again. However, this may not be true for all cases. The table can be on the book, for some reason; what about "Laura owns the book"? The relation between Laura and the book is owning, which has a converse "belonging to." Then, the state can be expressed as "The book belongs to Laura." When these two relations are the same, it would be legitimate to express the state as "The book owns Laura," which is nonsensical. There cannot be such a state. Then we cannot show symmetric relations upon our assumption. The converse of the assumption, if the same state arises from the holding either R and $R*$, then these relations are the same, and R is symmetric, is also false for the same reasons. Then, how can we determine non-symmetric relations? What about that a relation does not have a direction when it does not have a converse, which expresses the same state? However, this is nothing but begging the question.

One may wonder why we do not just claim that aRb is the same as bRa, then R is symmetric. For instance, if a is identical with b, then b is identical with a, therefore *being identical* is a symmetric relation. Even one can say that when we take the relation as a relational property and true for both relata, the relation is symmetric. "The major is meeting her vice-president" is another example of this. When we change the relation into *is loving*, we cannot claim that

is loving is a symmetric relation.[11] There will always be some indeterminacy, whether semantic or metaphysical when we convey the direction of relations.[12]

Philosophers have invested in finding a solution that offers a distinctive feature for non-symmetric relations without referring to direction. Here we will speak of only two of them by following MacBride (2020).[13] The first solution, positionalism, acknowledges that the distinctive feature of a non-symmetric relation R with respect to its relata can be explained "locally" in terms of R and the relata. In positionalism, the argument positions are conceived as entities without having an intrinsic order. That is, they cannot determine any significant order or direction; rather, the relations differ only with respect to the positions of the relata. For instance, a relatum can be either the subject or the object of a relation, say *to love*; so the occupants of the argument positions and the relation characterize a non-symmetric relation. On the other hand, primitivism, the second solution, does not admit the recognition of the relata: All relations are primitive, so they are unanalyzable. Moreover, the difference between the states aRb and bRa are taken primitive as well (ibid.).[14]

The category of relations contains complications even for humans' understanding. So, we need to end this section with too much puzzlement at hand: do relations depend upon the existence of substances that bear them? Or, on the contrary, are they ontologically independent entities? How can we speak of the intrinsic natures of the entities which pave the way for the existence of relations? How should we exhibit the nature of n-adic relations? How can we differentiate the relation of "cause" is transitive in one context and intransitive in another? When we admit relations as universals, should we abandon the view that all relations are primitive -since there is a hierarchy in universals- or should we re-

[11] Further, its converse is 'is loved by,' which is formally different.

[12] For semantic and metaphysical indeterminacies see Taylor and Burgess (2015).

[13] For instance, anti-positionalism is a strategy that rejects the distinctive feature of a non-symmetric relation explained locally.

[14] Whether we should annotate "is meeting" as a non-symmetric relation is a perplexity. If the relata are humans, then *is meeting* symmetric; however, consider "the principle meets the requirements," the very same relation is non-symmetric. A question of which approach to relations is better for knowledge representation in unstructured data is kept in mind.

ject hierarchy in relations? And above all: How should we represent relations in the machine, keeping the option of rejecting the existence of such "ontologically weird" things in toto, as Lowe (2016, p. 111) suggests?

C.5 Properties, Predicates, Relations

Can properties, predicates, and relations be equalized? Let us start with predicates and properties. Although predicate and property are used interchangeably, properties are articulated by predicates; that is to say, predicates are *linguistic entities*, not ontological ones (Orilia & Paolini Paoletti, 2020). Predicates are verbal phrases, like "is sweet" or "eats," predicates can be nominalized by suffixes –ity or-ness, or via gerundive or infinitive phrases. "Sweetness," then, can be taken as a property.[15] Furthermore, properties are semantic values of predicates for a kind of nominalism, yet they are more than being a value, such as accounting for similarity (Rodriguez-Pereyra, 2019). Being a semantic value is a role of properties, so predicates have their own ontological status. In the same vein, all properties are truly predicates of an entity (G. F. Stout, 1940, p. 117), so that we can think of predicates as properties. In other words, a predicate can only be a property (Orilia & Paolini Paoletti, 2017).

G. F. Stout (1940, p. 120) highlights that the source of our knowledge of an entity is not solely our observations of the entity itself. Moreover, the states that the entity has relations with other entities are its properties as well. As we uttered above, relational properties are properties that an entity share with another entity simultaneously. Thus, relations can be thought of as a kind of property. A question arises: what is the connection with predicates and relations? Should we take predicates as ontological entities? When we follow Findlay (1936), we should conclude that since relations are universal, predicates can be regarded as limiting cases of relations. However, are relations commit-

[15] According to Orilia and Paolini Paoletti (2017), 'sweet' and 'sweetness' distinction is not ontological; instead, it is a grammatical issue. In English, we say "Baklava is sweet," not "Baklava is sweetness." However, we cannot rule out this grammatical distinction. In formal settings, we typify words, and possible relations and possible combinations occur with respect to the part of speech tags. We will come back to this problem when we discuss the relation between formalism and metaphysics.

ting one-over-many? Let us leave these questions for a while and continue our categorical investigation.

C.6 The Role of Time and Space

Please think of the books of İhsan Oktay Anar and his writing the books. There was a moment when Anar was born; there was a period when he was writing one of his works up; there has been a moment since that book has been accessible. We can legitimately take a published book of Anar as a particular entity/an individual; at that rate, can we still legitimately regard his writing a book as an individual? Suppose that we have two different time instances, t_1 and t_2, when his book, B_1, has been published before t_1, and he has been writing his book, B_2, between t_1 and t_2. Take any n, such that $t_1 < n < t_2$. While B_1 is the same for any n, B_2, keeping its identity continues to change. Then, B_1 and B_2 should have different ontological statuses. In ontological parlance, the entities that are present wholly at every time instances when they exist are called endurants; the entities that happen through time and have time intervals are called perdurants (Cf. Yargan, 2020b). The latter is separated from the former by having temporal parts, so the individuals must be separated from events, which are obviously perdurants.[16] There are no perdurants in Aristotle's categories; there are no endurants in Whitehead's categories. Should we accept or reject the categorical distinction between individuals and events when representing Anar's books and writing one of his books, or a cell and crossing-over in a cell, for a machine?

Now think of a vase and the clay of which the vase is made. This typical ontological example is to question whether the different entities should be separately represented when they occupy the same space at a given time. The vase and the piece of clay exist in the same space and time, so it is legitimate to represent them separately since they have different ontological statuses: the vase is ontologically dependent on some amount of clay. However, Aristotelian hylomorphism rejects such representation since there is clay as a material cause of the vase. There is

[16] In the literature, continuant and occurrent are used interchangeably with endurant and perdurant, respectively. However, this is not a consensus. See Satıoğlu [Yargan] (2015).

a vase form that is the formal cause of the clay: the form of the clay is the vase, the matter of the vase is clay. They are one and the same ontological entities in space and time, yet they are different when analyzed hylomorphically. Hence, should we teach some hylomorphic analysis to the machine?

C.7 Tropes

We think that most of the ontological categories boil down to two dichotomies: universal-particular and substance-property and that there are two main relations: inherence and dependency. We have mentioned myriad views on combining these categories and relations, each of which has its own superiorities and deficiencies in representing the world. In the following, we will introduce the last ontological category, which is no exception of having superiorities and deficiencies: tropes.

Regarded as the father of tropes, Williams (1953) takes them as constituents of all of the entities. He develops his idea of how partial similarity and difference are both subsumed and inherited in entities (p. 4). He pictures three lollipops with the following features: the first lollipop has a red round peppermint head, the second one has a brown round chocolate head, and the third has a red square peppermint head. These three lollipops have one exactly similar part: their sticks. The proposal behind this claim is that when two entities are partially similar to each other, then a part of one entity is "wholly or completely similar" to a part of the other one. Ignoring the spatial aspects, for now, we can claim that each lollipop is partially similar to each other with respect to a physical part, or a "gross part," viz., stick. Williams (1953) proposes that we should treat "fine parts," namely, abstract components (such as shape, color, flavor), in precisely the same way: the first and the second lollipops are partially similar with respect to their shape; the first and the third lollipops are partially similar with respect to their color and flavor. Indeed, he purports that we can speak of whole similarity and partial similarity and dissimilarity of both gross and fine parts (p. 5); that is how we know of complex wholes, namely entities (p. 6). To make an example of tropes, consider the particular shape, color, or weight of an individual entity. Two lollipops "share" a property of shade of red, yet

each exemplifies a set of redness-trope. Since each redness-trope is numerically distinct, there must be at least two redness-tropes; but at the same time, they resemble each other.

The crucial point of Williams' doctrine lies in his interpretation of the term "abstract." As mentioned above, there are myriad interpretations of abstract. Williams (1953, p. 15) uses *abstract* to mean "*partial*, *incomplete*, or *fragmentary*, the trait of what is less than its including whole."[17] That is, what is an entity is a total that is the composition of abstract parts. For instance, a lollipop as a whole has constituents its color, its stick, its shape, its flavor, its weight, and alike. Moreover, he disagrees with taking abstract as universal and argues that the abstract components must be particular. Williams (1953, p. 10) avers that there is no ontological distinction between concrete particulars and abstract universals. Universals are abstract nouns that refer to sets of particular cases abstract parts. For instance, the redness of the lollipop stands for its particular case of Redness. Thus,Williams (1953, p. 7) defines *abstract particulars* as tropes, which are particular entities, which are "either abstract or consisting of one or more concreta in combinations with an abstractum" (p. 7). In that vein, tropes are *the very alphabet of being*.

Accepting "abstract particulars" does rule out neither universals nor concreteness (p. 16). Universals are sets of tropes, where a set is "a *class* of which the terms are members; and particulars are sums of tropes, where a sum is "a whole of which the terms are parts" (p. 9; italics in the original).[18] We know concrete things from their abstracta (in the sense of *part of the whole*), not from the bulk without properties (p. 16). The first things we realize when perceiving an entity, say a dress is its color, its style, its fabric, its sleeve length, its collar style, and alike. Namely, everything from pain to lollipops, from the Sun to music, can be constructed out of tropes. In sum, the trope provides one "rubric which is

[17] As Rojek (2008, p. 361) defines *concrete* as "a knitted whole."

[18] An utmost attention should be given to the fact that there is nothing as abstract universal in any trope theory.

hospitable to a hundred sorts of entity which neither philosophy, science, nor common sense can forego" (p. 17).

Said all, starting the history of the theory of tropes from Williams would be a mistake. Maurin (2018) claims that this theory could date back to Aristotle. Several names offered alternatives to *tropes*, throughout the history of trope theory, such as abstract particular, mode, moment, quality instance, concrete property, particularly property, unit property, intrinsic character, or quality.[19] Moreover, philosophers disagree on not only different labels but also the ontological status of the category of tropes.

We can classify trope theories into two (Rojek (2008, p. 366), Cf. (Maurin, 2018)). The foremost declaration in the first group is that tropes are the elements of being; whereas, that in the second group is that all properties are particulars. Let us examine them in turn.

C.7.1 Tropes: The Elements of Being

According to this Trope Theory, tropes are the only ontological category (Williams (1953), Maurin (2002), Campbell (1990)). The very duty of trope-only theorists is to show how other categories are made up of tropes.

Tropes, the single type of entity, are *abstract particulars*. They are not numerically identical and multiply located in space and time as universals are. For instance, think of two pieces of baklava. Baklava is sweet and sweetness can be abstracted from it. The sweetness of the first is different from that of the second. Each piece has its own sweetness, yet the two resemble each other concerning their taste. Trope-Only theories, on the other hand, do not take abstract as a non-spatiotemporal category (See, for instance, Williams (1953) and Campbell (1990)). The demarcation between abstract and concrete can lie in monopolizing the locations. Concretes monopolize their locations, whereas abstracts occur

[19] See Niiniluoto (2012) and Maurin (2018) for the philosophers who coined those terms. The last two belong to Paul (2017).

in conjunction with other things (Campbell, 1990). Or, abstracts can be taken as part or fragmentary (Cf. Williams, 1953). In one way or another, abstracts cannot be considered as universals in this kind of trope theory.

G. F. Stout (1940, p. 117) states that a "thing as a subject of predicates is simply identical with the complex whole including all properties truly predicable of it." what is *this* orange? G. F. Stout (1940) says that asserting its roundness and juiciness provides partial answers; thus, we would have a complete answer as long as we assign all its predictable properties. One may claim that asserting all the predictable properties cannot *fully* define an entity. Let us change the question of "What is *this* entity?" into that of "What is this entity as a subject of predicable properties?" A tenable answer has value in metaphysics, yet representing such an obscure "thing" in the machine is irrelevant. Since we do not represent somewhat, we "do not know what," rather what we perceive or think of a particular entity (G. F. Stout, 1940, p. 118). Moreover, things are the character complexes, and the character predicate of a thing is nothing but a constituent of the thing (p. 120). Once we follow this line of thought, a particular entity is "the whole complex including all its properties in their union with each other" (p. 118).

Other types of entities can be accounted for in terms of tropes. In the sense of an individual entity, such as a star or an electron, a particular is held to be a *compresent collection of tropes* or, as for Williams, a *sum of the tropes*. A universal, such as sweetness or Sweetness, is *a set of exactly similar tropes*. The sum-set notions are special in this theory. "Set" does not have any set-theoretic ontological connotations; it is just used to refer to similar tropes. "Socrates is wise" means that "the concurrence sum (*Socrates*) includes a trope which is a member of the similarity set (*Wisdom*)."

Certainly, there are too many metaphysical questions to be addressed. For instance, how can we differentiate two collections of the same compresent tropes? Is the relation of compresense among tropes the same as the relation between complex universals, such as the set of exactly similar particular cases of redness and oblongness? Is the relation of resemblance strong enough to rule out any substance-theoretical categories? How should the relation of identity be defined

in terms of tropes? Although so many others can be added, we should ask questions concerning a possible depiction of the world in the machine.

C.7.2 Tropes with Universals

Most of the one-category ontologists, if not all, are nominalists who reject the existence of universal properties. On the other hand, trope theorists need not be nominalists. They can reject universals as common properties, yet they can accept universals either as determinables or as concrete. That is to say, there are realist-trope theorists as well.

In the Several Senses of Properties part, we spoke of three kinds of universals. Recall that Rojek (2008) distinguishes universals as *properties* from universals as *aspects*. This distinction is based on two relations of inherence and determination. Properties *inhere* in entities, whereas aspects are *determined* by the properties. In this sense, all properties are determinate, and all aspects are determinable. Following that, one can exemplify this claim as sweet is a property, and taste is an aspect. On the other hand, recall that Rojek (2008, p. 370) avers that a *property* is "proper" to an entity; namely, it is something that belongs to an entity. Therefore, properties are particulars, yet tropes and other entities determine aspects. An aspect like sweetness is determined by a trope; an aspect like humanity is determined by an entity, such as Socrates. A sweetness inheres in cannoli, and a humanity is determined by Soctares, a bundle of tropes. Given that all properties are particulars and all properties are tropes, we can conclude that there are universals as aspects. By declaring the existence of determinable universals, not all of the trope theories can be monopolized by nominalists.[20]

All trope theorists reject the existence of abstract universals, but some of them accept *concrete universals*. Recall that Rojek (2008) speaks of concrete universals as the third kind of universals. He suggests that concrete universals are Aristotle's second substances or Wittgenstein's "family of resemblances" (p. 373). Once again, recall that what demarcates universals and particulars is the

[20] See Niiniluoto (1999) for further discussions.

primitive relation of inherence: universals are not properties but *wholes* which encompass their instances, properties, considered as abstract universals, on the other hand, are particulars. For instance, abstract sweetness inheres in many distinct desserts, whereas concrete sweetness is a *whole* constructed by all the desserts that ever have been and will be. Thus, as tropes are particulars, there is another category of being other than tropes. So, it is legitimate to declare a trope theory with concrete universals.[21]

C.7.3 Tropes and Concrete Objects: Together or The Same

In this appendix, we have been investigating various categories of being, and we have experienced that the meaning of the categories varies from one philosopher to another. In this line, the following is inevitable: that Williams defines tropes as abstract particulars, and that "abstract" means what is partial or incomplete is untenable for many trope theorists. Some think that tropes are concrete (e.g., Giberman); for some, tropes are both abstract, and concrete (e.g., Simons), or some think tropes are a kind of property (e.g., Mulligan) (Maurin, 2018).

We claimed that all trope theorists acknowledge that all properties are tropes. Does this statement legitimately follow that tropes are properties? Williams would refuse such a claim since a trope-only theorist has to maintain tropes as objects, or at least *a kind of* object. As Williams (1953, p. 5) states, we christen tropes: we name such particulars. On the other hand, if all properties are tropes, and all properties are independent entities, then tropes are a kind of dependent entities, contrary to Williams and Campbell. This situation can be observed linguistically: Tropes express the ways entities are. What about simply accepting that tropes are neither objects nor properties and that they are the independent building blocks of every entity? Trope compositions can construct objects and properties. Or, consider that 'being sharable' is necessary for 'being a property' and that monopolizing a spatiotemporal location is characteristic of

[21] According to the theory of concrete universals, concrete particulars and concrete universals are one and the same. The idea behind this is that "concrete" means things are united/knitted. Within the limits of this work, we do not go through it.

objects; tropes are neither properties nor objects due to their being independent and particular, respectively. Following these, tropes can be assumed to be instantiated in individuals, which amounts to substance-trope view (Cf. Simons, 1994).

The perplexity of tropes being either property or an object springs from the perplexities of abstract-object-property-universal notions and their connotations. We can discuss: If tropes are a kind of objects, they are "non-transferable," but when they are a kind of property, they are "transferable" since properties are instantiated, but objects are not. That means tropes are not abstract, then not universal. On the other hand, we can discuss the other way around, such that tropes are properties and abstract. However, we should never ever forget that all our discussions are done for the sake of the realization of machine understandability. It would be wise to reconsider the connotations of these categories for machine ontology.

C.7.4 Further Conflicts among Trope Theorists

The confusions about the nature of tropes are not limited to the property-object dichotomy. As we emphasized before, tropes can be categorized as primitive, simple, and complex. Tropes are primitive only if they have no constituents (e.g., Campbell (1990), Maurin (2002)). Tropes are simple either when they have no parts or when they are made up from other tropes. Obviously, the trope-only theorists, tropes, the very alphabet of entities, are primitive or ontologically simple. Lastly, tropes are complex only if they have internal constitutions, which are either intrinsic aspects of a trope or some other ontological category other than tropes. The relation of resemblance instantiates an intrinsic aspect, for instance, as it helps us compare tropes. That is, a trope must have its intrinsic properties (Alvarado, 2019). On the other hand, if an internal constitution is of not tropes alone, then the trope theorists maintain either a trope-universal theory or a trope-substance theory.

Rojek (2008, p. 364) distinguishes two kinds of simplicity. The first kind of simplicity is necessary for any trope theory, whereas the second kind, which

is the primitiveness of tropes, is unnecessary. As nothing inheres in a trope, tropes are simple by their nature; yet there can be trope constituents. That is, tropes are simple in respect to inherence, and they can be complex regarding other relations, such as in respect to determination. On the other hand, Molnar (2003, p. 37) notes that modes, i.e., tropes, are neither simple nor complex. He claims that tropes neither have parts nor have themselves as parts, so that they are just out of any mereological study (p. 44).

Another conflict among trope theorists is how to individuate tropes. For some of the trope-only theorists, two tropes are distinct if and only if they are numerically distinct; since they are primitive. Calling it the *Primitivist Principle*, Ehring (2011, p. 76) such an approach requires no analysis or reduction to individualization of tropes. Campbell (1990, p. 69) depicts this principle as "individuation is basic and unanalyzable" since a trope is just what it is. Since there is neither ontological analysis nor metaphysical explanation is possible concerning this principle, there is the *Spatiotemporal Principle*, which explains necessary and sufficient reasons for the individuation of exactly two similar tropes: two tropes are distinct if and only if either they do not exactly resemble or they are distinct from each other spatiotemporally (Ehring, 2011, p. 76). However, this principle cannot explain mutually excluded tropes. A roundness trope and a squareness trope are exactly similar with respect to their both being shape, yet they cannot have the same spatiotemporal location. Thus, this principle fails in explaining same-level trope individuation. To solve this problem, another individuation principle can be offered: the *Object Principle*. Tropes are individuated with reference to the entities that "contain" them. Two exactly similar tropes are identical only if the entities that they constitute are exactly identical. This principle, however, makes individuation circular (Maurin, 2018). As an entity is a trope composition, it is known by the individuated tropes. Next, we operate

this trope composition in order to distinguish the tropes: we know of an entity via its tropes and a trope via the entity.[22]

C.8 Conclusion

This appendix studies how ontological categories, particularly relations and properties, are studied in philosophy. Its purpose is to investigate philontological characteristics of properties and other categories directly related to them. Additionally, it is witnessed that there are so many philosophical disputes, such as in trope theory: Hochberg and Armstrong reject simple tropes; Ehring accepts simple tropes from a different perspective; Molnar rejects both aspects, as the two tropes are not simple due to their internal relations. All these disagreements are of philontologies; however, our aim has never been pondering these issues. Instead, it is to find the most suitable guiding categorical theory for a machine ontology in light of this work.

[22] The individuation of tropes has been discussed from other perspectives, such as *swapping*, *sliding*, and *piling* arguments. Here we neither continue the issue of trope individuation nor define all these arguments; in which anyone interested are encouraged to see Ehring (2011, pp. 78–91) for details.

D. CURRICULUM VITAE

Contact Information

dilek.yargan@gmail.com

Research Interests

Data Science, Formal Ontologies, Philosophy of Artificial Intelligence, Philosophy of Computation, Mathematical and Philosophical Logics, Ancient Greek Philosophy, Philosophical Theology

Education

PhD: Philosophy, METU (2022)

Dissertation Title: A Computational Ontological Model for Machine-Understandable Data in Artificial Intelligence

MA: Philosophy, METU (2015)

Thesis Title: A Philosophical Approach to Upper-Level Ontologies

MS: Secondary Science and Mathematics Education, METU (2008)

Thesis Project: High School Students' Beliefs About Mathematics

BS: Mathematics, METU (2005)

Academic Works

Yargan, D., 2022, Ulusal Sağlık Veri Sözlüğünün Ontolojik İncelemesi [An Ontological Review of National Health Data Dictionary]" in *Tıp Bilişimi III*, İstanbul University Press, *in press*

Yargan, D. and Zambak, A. F., 2022, Medical Ontologies in *Tıp Bilişimi II*, İstanbul University Press, *in press*

Yargan, D., 2021, Ayhan ile Eren Erdemin Peşinde: Aristoteles'in Erdem Ahlakı Işığında Çocuk ve Etik [Ayhan and Eren in Pursuit of Virtue: Child and Ethics in Light of Aristotelian Virtue Ethics], *Arkhe-Logos*, 11 (2), pp. 115–124.

Yargan, D., 2020, Betimleyici Mantıklara Giriş [An Introduction to De-

scription Logics], *Beytulhikme*, 10 (4), pp. 1303–1324.

Yargan, D. and Zambak, A. F., 2020, Medikal Ontolojiler [Medical Ontologies], in *Tıp Bilişimi*, İstanbul University Press, E-ISBN: 978-605-07-0773-1, pp. 25–60.

Yargan, D., 2020, Is Science Without Explanations Possible?, *Arkhe-Logos*, 9 (2), pp. 109–122.

Yargan, D., 2020, Üst Düzey Ontoloji İnşasındaki Felsefi Yaklaşımlar [Philosophical Aspects of Building Upper-Level Ontologies], *Kilikya Journal of Philosophy*, (1), pp. 32–50.

Yargan, D., 2019, Formel Ontolojiler ve Betimleyici Mantıklar [Formal Ontologies and Description Logics], *Archives of Philosophy*, 51, pp. 271–281.

Conferences, Published Proceedings, Lectures

"Bilinç İşlevden Ayrılamaz", M.A. Cohen & D. C. Dennett [Consciousness cannot be Separated From Function], Translator, in *Bilinç: Çağdaş Bir Antoloji*, M. Doğan (ed.), *in press*

"Ders 9: Kadının Konuşma ve Politik Katılım Hakkı", March 10, 2022, Lecturer, in *Kadınlar için Felsefe Okulu* [School of Philosophy for Women], March 8-10, 2022, Aristotelian Society and Muğla Municipality

"Şeytanın Avukatlığı: Gerçekleştirilemeyecek Vaatler" [Devil's Advocate: Unrealizable Promises], in *AI in Social Sciences* session, September 1, 2021, Artificial Intelligence Summer School, September 1-3, 2021, Istanbul, IU-Cerrahpasa, Kocaeli, Sakarya, and Yalova Universities

"Knowledge Standardization in the ESBS: Why do we need a reference ontology?", January 16, 2021, Invited Speaker, *Method, Research, Theory Programs Project*, Boğaziçi University

"Towards Building ESBS Ontologies with Protégé 5.5", May 25, May 30, and June 1, 2021, Lecturer, Method, Research, Theory Programs Project, TÜBİTAK Project no: 118C257, Boğaziçi University

Smart Cities Project, December 2020- February 2021, Smart Cities Terminology Editor and Controller, Ministry of Environment

"Tıpta *Linguae Francae*: Medikal Ontolojiler" [Linguae Francae in Medicine: Medical Ontologies], with Aziz F. Zambak, From the First Computers to Medical

Informatics: 60 Years of Adventure, in Digital Health and Artificial Intelligence Applications in Medicine section, September 30, 2020, e-workshop

"Yapay Zeka ve Formel Ontoloji" [Artificial Intelligence and Formal Ontology], October 26-27, 2019, Lecturer with Aziz F. Zambak, Computer Science Application and Research Center, Department of Informatics, İstanbul University, İstanbul

"Is Science Without Explanations Possible?", 2nd International Congress of Women Philosophers, November 14-16, 2019, Muğla

"Betimleyici Mantık" [Description Logics], August 8, 2019, Lecturer, Logic Summer School, Feza Gürsey Center for Physics and Mathematics, Turkish Logic Society

"Bilgi Hiyerarşisini Dümdüz Eden Büyük Veri" [Big Data Flattening the DIKW Pyramid], 9th Workshop on Logic, April 18-19, 2019, Mardin

"Büyük Veride Keşif ya da Kaşifçilik" [Alleged Discovery in Big Data], 8th Workshop on Logic, May 9-11, 2018, Zonguldak. ISBN: 978-605-66311-4-6, pp. 505–512.

" 'Logical' Concerns in Data Science", Istanbul International Congress on Philosophy, May 2-4, 2018, İstanbul. ISBN: 978-605-80953-0-4, pp. 199–206.

"Formel Ontolojilerin Esnekliği ve Dinamikliği Üzerine" [On Dynamic and Flexible Formal Ontologies], 7th Workshop on Logic, June 29-30, 2017, Samsun. ISBN: 978-605-66311-2-2, pp. 697–703.

"A Philosophical Introduction to Upper-Level Ontologies", 2nd International Symposium on Philosophy, Education, Art & History of Science, May 3-7, 2017, Muğla, p. 514.

"What Would It Like to Be Visiting an Art Gallery with Wittgenstein?", 2nd International Symposium on Philosophy, Education, Art & History of Science, May 3-7, 2017, Muğla, p. 515.

"A Taxonomy of Questions with Respect to the Universal-Particular Dichotomy, 9th International Conference on Formal Ontology in Information Systems, 2016, Annacy France, `http://ceur-ws.org/Vol-1660/ecs-paper4.pdf`.

"Süreler Mantığının Formel Temelleri ve Süreç Metafiziğine Etkileri" [The Formal Bases of Temporal Logic and The Theory of Process], 6th Workshop on Logic, May 26-27, 2016, Artvin. ISBN: 978-605-66311-1-5, pp. 353-361.

Experience

Data Scientist; Functor, Ankara (February 2022 – present)

Data Scientist; Endeksa, Ankara (June 2018–January 2022)

Research Associate; Laboratory for Computational Ontology, METU, Ankara (August 2014–August 2022)

Intern Teacher; Ballard Community School District, Huxley, Iowa (January–March 2008)

Freelance Tutor, Mentor, NLP Practitioner (2004–2017)

Research Assistance, Registrar's Office, METU, Ankara (September 2005–September 2009)

Awards, Certificates, International Exchange

METU Course Performance Award, 2012-2013 Academic Year, Graduate School Social Sciences

Erasmus Student Exchange Programme September 2013-March 2014, Johannes Gutenberg-Universitaet Mainz

DAAD- University Summer School Scholarship, August 5-30, 2013, Ruprecht-Karls-Universitaaet Heidelberg

Fulbright Teacher Internship Scholarship January-March, 2008, Iowa State University

METU Course Performance Award 2006-2007 Academic Year, Graduate School of Natural and Applied Sciences

Erasmus Staff Mobility Programme, June 22-26, 2009, Katholieke Universiteit Leuven, Belgium

Italian Language Diploma, 2003, TÖMER, Ankara University

Languages and Skills

Turkish, English, German, Italian, Latin, Spanish, Ancient Greek

R, Python, Protégé, LaTeX, SQL, SPAQRL, Agda

E. TURKISH SUMMARY/TÜRKÇE ÖZET

E.1 Giriş

Büyük Veri'nin insanlık tarihi boyunca dönüşümler sağladığı/sağlayacağı inancı onu yüzyılın en önemli olgularından biri haline getirmiştir. Ancak insanlık tarihini şekillendiren üç alan –endüstri, bilim ve Web– incelendiğinde, Büyük Veri mevcut teknolojilerle bu alanlarda gerçek bir dönüşüm sağlayamaz. Çünkü, bu alanlarda Büyük Veri'nin dönüşüm sağlayabilmesi için makinenin otonom olması, diğer bir deyişle ilk önce Büyük Veri'yi anlayabilen ve işleyebilen bir makinenin inşası gereklidir. Bu tez, makine-anlaşılabilirliğinin felsefi ve berimsel ilkelerini tanımlamayı, makine-anlaşılabilirliğini sağlayacak bir veri işleme yapısını keşfetmeyi, buradan hareketle, verilerin nasıl temsil edilmesi gerektiğini araştırmayı ve üzerinde anlaşmaya varılan temsil sisteminin biçimselleştirilmesini sağlayacak bir araç bulmayı hedeflemektedir.

E.2 Büyük Veri, Endüstri, Bilim ve Web

Dünyanın üzerinde ilerleme kaydettiği en temel çarklar bilim ve endüstridir. Bu alanlardaki gelişmeler, dünyanın kaderini şekillendiren yeni dönemleri başlatmıştır. Yirmibirinci yüzyılda, donanım ve yazılımdaki gelişmelere ek olarak Büyük Veri'nin etkisini ile endüstriyel ve bilimsel üretim şekillerinde değişiklikler yaşanmaktadır. Hem endüstri hem bilim hem de yüzyılın en kritik bileşeni olan Web de Büyük Veri'nin dönüştürücü etkisi altındadır. O halde, bu alanlardaki veri tufanının neleri değiştirdiğini ve Büyük Veri'nin etkisinin nasıl bir devrim yaratabileceğini tartışmadan önce, Büyük Veri'nin ne olduğu incelenmelidir.

E.2.1 Büyük Veri

'Veri' sözcüğü günümüzün en çekici ve bir o kadar da aldatıcı terimi haline gelmiştir. Popüler makaleler ve kitaplar, veriyi en değerli kaynak anlamında "dijital çağın yeni petrolü" olarak tanımlar (Leonelli, 2014). Verinin bu kadar değer kazanmasının asıl nedeni veri hacmindeki üstel artıştır. 2017'de günde 2,5 kentilyon bit veri üretilmiş ve küresel internet nüfusu 2017'den Temmuz 2021'e kadar %36,84 arttığından üretilen veri sayısı önemli ölçüde artmıştır.[1] O halde, üretilen ve saklanan verilerin hacmi, hızı ve çeşitliliğindeki artış veriye 'Büyük Veri' adını koymayı gerektirmiştir.

Büyük Veri'nin çeşitliliğinden bahsederken üç veri çeşidinden bahsetmek gerekir. İlki yapısal veridir. Yapısal veri, en basit haliyle, makinenin işlemesine uygun hale getirilmiş veri demektir. Yarı-yapısal verinin ise yapısallaştırılmamış kısımları bulunur. Bunlar üzerinde çeşitli istatistiksel modeller kullanılır ya da yapısallaştırılır. Yapısallaştırılmamış veri ise verinin doğrudan kullanıma hazır olmayan çeşididir. Veri işleme yöntemlerinin bu veri yığınlarında düzgün çalışabilmesi için bu verilerin kısmen ya da tamamen yapısallaştırılması gereklidir. Büyük Veri'nin en önemli özelliklerinden biri %90'ından fazlasının yapısallaştırılmamış olmasıdır. Geleneksel veri yönetimi teknikleri yapısallaştırılmış veya yarı yapısallaştırılmış veri kümeleri üzerinde çalışabildiğinden, Büyük Veri teknolojilerinin yapısallaştırılmamış veriyi işleyecek biçimde geliştirilmesi son derece elzemdir. Sonuç olarak, veri biliminin ana konusu olan yapısallaştırılmamış olanlar başta olmak üzere Büyük Veri'nin her türlü veriyi yönetebilmesi için yeni veri teknikleri geliştirilmeli ve yeni anlayışlar ortaya atılmalıdır.

E.2.2 Endüstri

"Endüstri" terimi, "mallar, hizmetler veya gelir kaynakları" üreten veya sağlayan ekonomik faaliyetlere işaret eder (Britannica, n.d.-b). Endüstri tarihinde sıçra-

[1] https://www.domo.com/learn/infographic/data-never-sleeps-9 adresinden güncel sayılara bakılabilir.

malar yeni teknolojiler, en son bilimsel keşifler ve/veya beklenmedik sosyal fenomenler (nüfustaki çarpıcı artış gibi) ile gerçekleşir. Bilişsel ve berimsel alanlardaki teknolojik gelişmeler sayesinde yeni bir sıçramadan, Endüstri 4.0'dan bahsedilmeye başlanmıştır.

Bu çalışmada endüstri, genel kanıyla da uyumlu olarak, dört aşama incelenmiştir. Tarih kitaplarında endüstride yaşanan ilk dönem ve devrim *Sanayi Devrimi* olarak anılır. Bu dönem, buharlı makinelerin kullanılmaya başlandığı ve hemen ardından insanların kas güçlerini özgürleştirdiği, bazı işlerin mekanize edildiği dönemdir. Piyasadaki artan taleplerin ve mekanikteki keşiflerin etkisiyle ikinci dönemin, *Endüstri 2.0*, tohumları atılır. Bu dönem, fabrikalarda elektrik kullanımı ve böylece seri üretimi gerektiren montaj hattı devreye girmesiyle başlar. Üçüncü dönem, *Endüstri 3.0*, fabrikalarda otomasyona yol açan bilgisayarların ortaya çıkmasıyla başlayan *Dijital* veya *Bilgisayar Devrimi* olarak adlandırılır. 2011 yılında Hannover Fuarı'nın açılışında ortaya çıkan "Endüstri 4.0" terimi ile merkezi olmayan ve otonom fabrikaları hedefleyen *Endüstri 4.0* olarak adlandırılan dördüncü endüstri devrimi içerisindeyiz.

E.2.2.1 Endüstri 4.0

İnternet teknolojileri, veri analiz teknikleri, Büyük Veri, robotik ve benzeri sayesinde üretim süreçlerinde gerçek ve tam otomasyon sağlanması hedeflenmektedir. Bir anlamda, bu tür gerçek ve tam otomasyon, *otonom* olarak adlandırılabilir. Bu tarz bir otomasyon, bilgi yoğun üretimin olduğu otonom bir endüstridir. Bu nedenle, Endüstri 3.0'dan Endüstri 4.0'a geçiş demek gömülü sistemlerden (merkezi üretimden) siber-fiziksel sistemler ağına (merkezi olmayan üretime) geçiştir. Endüstri 4.0, hammadde elde etmekten müşteri memnuniyetini karşılayacak lojistiğe kadar tüm süreçleri yöneten merkezi olmayan endüstri dönemidir. Böylece, kâr oranlarının arttırılması, maliyetlerin düşürülmesi, müşteri deneyiminin iyileştirilmesi, piyasaya yeni sürülen hammaddelerin korunması, yaşam boyu değerin optimize edilmesi ve diğer pazar konuları kaçınılmaz olarak otonom makineler ile gerçekleşebilir. Çünkü, merkezi olmayan bir üretim, tüm süreçlerin dinamiklerini aynı anda değerlendirmeyi, verilerden çıkarımda bulunmayı, tavsiye etmeyi, karar vermeyi gerektirir. Hacmi, çeşitliliği ve hızı oldukça fazla

olan verileri işleyerek pazarlamanın tüm yönlerinde, yani ürünlerin tasarımında, üretiminde, işletiminde ve hizmetinde özerk karar verebilen makinelerin var olmasından sonra Endüstri 4.0'dan bahsetmek mümkün olabilir.

E.2.3 Bilim

Dünyanın gidişatına yön veren en yüksek insani girişimlerden biri olan bilim, fenomenlerin sırlarını anlama yöntemlerine sahip karmaşık bir sistemdir. Özellikleri sistematik gözlem, deney, akıl yürütme, hipotez ve teorilerin inşası ve testtir. Veri toplama, doğal veya laboratuvar ortamında veya simülasyonlardan yapılabilir; akıl yürütme yöntemleri farklı bilimsel çalışmalarda farklılık gösterebilir; hipotezleri test etme yolu değişebilir, ancak bilimin amacı olan bilgi üretimi faaliyeti değişmez.

Bilimsel ve/veya felsefi bakış açıları veya farklı yaklaşımlar bilim tarihinin farklı aşamalara veya dönemlere ayrılmasına neden olurlar. Bu çalışmada ise bilim tarihiyle, makinelerin bilimsel bilgi üretimine katkısı açısından ilgileniyoruz. Bu bakımdan Bilim 1.0'ı bilim insanlarının pasif gözlemciler olduğu, deneylerin *in vivo*'da yapıldığı ve bilimsel bilginin tümdengelimli akıl yürütme ile üretildiği dönem olarak belirliyoruz. Bilim 2.0 ise gözlemlenemeyenlerin görünür hale gelmesini sağlayan deney aletlerinin ve ölçüm araçlarının icat edildiği ve böylece toplanan verilerin hacminin arttığı dönemdir. Böylece, Bilim 2.0'daki bilim insanları deneylerini hem *in vivo*'da hem de *in vitro*'da yapabilir hâle gelmişlerdir. Bilim 3.0 ise bilgisayarların icadından sonraki döneme işaret eder. Bu dönemde, bilim insanları deneylerini yapmak için makineye bağımlıdır, çünkü veri toplama, veri işleme, deney ortamı oluşturma gibi işlemlerin gerçekleştirilmesi için makinelerin kullanılması zorunludur. Böylece, bilim insanları bilgisayar simülasyonları aracılığıyla bilimsel deneylerin kapsamına *in silico*'yu da katmışlardır. Bilim 4.0'a kadar olan tüm dönemlerin ortak özelliği bilimsel bilgi üretiminin insana ait olmasıdır. Bilim 4.0 ise veri tufanındaki unsurları insanların bir araya getiremeyeceğini vurgulayarak bilimsel bilgi üretimine makineleri de katan dönem olacaktır.

E.2.3.1 Bilim 4.0

Deney ve ölçüm teknolojisindeki gelişmeler sonucunda genellikle veri tufanı olarak adlandırılan büyük miktarlarda veri ortaya çıkmıştır. Araştırmacılar, bu kadar büyük miktarda veriyi işleyerek bilim yapabilmeyi sağlayan bir çerçeve sunabilecek yeni yöntemler aramaktadır. Bu çerçeveye *eBilim* veya *veriye dayalı bilim* denir ve yöntemler *veri analitiği*ni oluştururlar. Veri tufanının tüm bilimlerin geleceğini oluşturduğu yadsınamaz ve buna göre bilim insanlarının veri analitiği becerileri kazanmaları gerekmektedir. Öte yandan, hesaplama gücündeki ve algoritmalardaki ve hiç bu kadar büyük olmayan verilerdeki geliştirmeler sadece Bilim 3.0'daki gelişmelerdir. Çünkü ne yapay sinir ağları ne Bayes çıkarımı ne de diğer makine öğrenme algoritmaları, yani hiçbir veri analitiği aracı, ürettikleri sonuçların hesabını veremez. Bilimde açıklama hayatidir, bu nedenle veri analizinin ürettiği hiçbir açıklama, bilimsel iddia anlamına gelemez. Ne zamanki makineler "neden" sorusunu yanıtlayabilir o zaman ürettikleri sonuçlara bilimsel keşifler olarak güvenebiliriz.

İstatistiksel modeller, araştırmacıların teorilerini geliştirmelerine yardımcı olmak için kullanılır. Araştırmacılara -eldeki araştırmayla hem doğrudan hem de dolaylı olarak ilgili tüm belgeleri okuyamayan insanlara- önerilerde bulunan makineyi yaratmayı hedefliyorsak, o zaman makine-araştırmacılar tarafından sağlanan içeriği anlayabilmelidir. Bu ancak bilimsel Büyük Veri'nin anlamsal yönleri analiz yöntemlerine dahil ettiğinde mümkün olabilir. Bilgisayar simülasyonları ve cihazlardan gelen veriler yapısallaştırılmış olmasına ve bilimin belirli alanlarında çok sayıda taksonomi ve ontoloji kullanılmasına rağmen, bilimsel Büyük Veri'yi yapısallaştırmak zor ve zorunlu bir iştir. Bu nedenle, verileri standartlaştırmanın yollarını bulmazsak, Büyük Veri bilimde asla devrim yaratamaz. Bunun gerçekleşmesi için makinenin veri iyileştirme yeteneğine sahip olması gerekir; yapısallaştırılmamış verileri otomatik olarak yapısallaştırılmış bir şekilde temsil edebilmelidir. Ayrıca, bilgisayar simülasyonları, deneyler ve bilimsel belgeler bilimin bir yönünü yansıttığından, makine açık dünya sistemleriyle başa çıkabilmelidir. Kısaca, Bilim 4.0 makinenin bilimsel bilgi üretiminde yer

aldığı döneme işaret eder ve bu dönemde makinelerin Büyük Veri'yi anlayabilmesi zorunludur.

E.2.4 Web

Tim Berners-Lee'nin hayali CERN'deki bilim insanlarının bilgi paylaşabileceği, projeleri takip edebileceği, eski projelerin teknik detaylarına ulaşabileceği veya kişisel bilgisayarlarda saklanan kayıtlı bilgileri bulabileceği bir 'uzay' yaratmaktı (Berners-Lee & Fischetti, 2001). Bu ilk küresel enformasyon alanı yaratma hayali 1989'da gerçekleşti ve adına *Dünya Çapında Ağ* ya da sadece *Web* denildi. Web, o zamandan bu yana günlük hayatımızın vazgeçilmez bir bileşenidir.

Tim Berners-Lee'nin hayal ettiği uzayın, yani bilgisayarlarda bulunan enformasyonun birbirine bağlı bilgisayarlar aracılığı ile herkesin ulaşabileceği şekilde tasarlanmış ağın ilk versiyonu Web 1.0'dır. Bu Web türünde tüm web sayfaları statiktir; kullanıcılar, sadece profesyonel içerik üreticileri tarafından oluşturulan içeriklere ulaşabilir ve okuyabilir. Web fikrinin sosyal etkileşimi gerektirdiğinden Web 2.0 ortaya çıkmış ve böylece insandan insana iletişim ve dinamik veri paylaşımı ortamı oluşturulmuştur. Böylece artık web içeriği de kullanıcılar tarafından üretilmeye başlanmıştır. Bununla birlikte, web içeriğinin hacmindeki önemli artış, Web'deki içeriğin anlamsal olarak birbirine bağlı biçimde düzenlenmesini gerektirdi. Web 3.0, makinenin Web üzerindeki verileri analiz etmek için içeriği okuyabildiği Web aşamasıdır. Bu dönemde Web'de bulunan veriler, insanların bağlamı gözeterek verilere atfettikleri anlamsal özelliklerle ile birbirine bağlanabiliyordu. Ancak, veri tufanından en çok etkilenen alan olan Web'de bağlama bağımlı anlamsal özellik atfetme sürecinin insan işi olmaktan çıkması, makinenin bunu otomatik olarak gerçekleştirmesi gerekmektedir. Bu da bizi Web 4.0'a getirecektir.

E.2.4.1 Web 4.0

Büyük Veri'nin en büyük etkisi Web üzerindedir. Web'de günlük olarak oluşturulan veriler petabitleri aşmakta, günde milyarlarca fotoğraf ve video yüklenmekte, milyarlarca yorum yayınlanmakta, milyonlarca web sayfası oluşturulmakta veya güncellenmekte ve binlerce belge yüklenmektedir. Arama motor-

ları, sorgudaki anahtar sözcüklere göre ilgili web sayfalarını bulmaya yardımcı olsalar da sonuçlar milyonlarca sayfanın içinde bir yerlerde olabilir. Bu nedenle, Web 4.0, arama alanını daraltan ve kullanıcıların tüm sayfaları gözden geçirmesine gerek kalmayacak şekilde sorguyu özelleştiren sınıflandırmalara olanak tanımalıdır.

Web 4.0'ın diğer bir özelliği de sorguları yanıtlamak olabilir. Web 4.0'ın alametifarikası aramaya değil, sorguya dayalı içerik üretmek olmalıdır. Herhangi bir sorgu süreci, mantıksal gereklilik ve dolayısıyla bir içerik üretimi gerektirir. Yani, makine sorguya göre farklı web sayfalarındaki enformasyonu birleştirip yeni sonuçlar üretebilme gücüne de sahip olabilir. Sonuç olarak, makinenin ilgili tüm kaynakları okuması, yorumlaması ve aramaya göre sınıflandırması ve daha sonra kümelenmiş web sayfalarında bağlam üretimi yapabilmesi gerekmektedir.

E.3 Etkin Eyleyici Makine

Bir önceki bölümde Büyük Veri'nin ne olduğu ve Endüstri, Bilim ve Web alanlarının Büyük Veri ile değişiminin nasıl olacağı/olması gerektiği tartıştık. Endüstrinin geleceği, makinenin ürün üretimlerden dağıtımlara kadar her aşamayı otomatikleştirdiği akıllı fabrikalardır. Bunun gerçekleşmesi için, insan-makine iletişiminde birlikte çalışabilirlik ve bilgi yoğun üretimde merkezi olmayan zekâ olması gerekir. Her cihazın aynı dilde konuşmasını sağlamak imkânsız olduğu için merkezi olmayan olmalıdır. Cihazların birbirini anlaması ve Büyük Veri'yi anlaması çok önemlidir. Bilim 4.0'da makine hem bilgi kaynağı hem de üretilen bilginin durumunu belirleyen eyleyici olacaktır. Bilimin geleceği, makinenin, adeta otonom bir iş arkadaşı gibi bilim insanlarına önerilerde bulunarak, soruları yanıtlayarak ve kalıpları açıklayarak bilimsel bilgi üretimine katkıda bulunmasını gerektirir. Öte yandan Web'in geleceği, makinenin içerik yöneticisi ve üreticisi olmasını gerektirir. İçerik yöneticisi olarak, makine herhangi bir konuda doğal dil sorgularına cevap verecektir; içerik üreticisi olarak, makine Web'deki hacimli enformasyondan bir sorgu üzerine içerik üretecektir. Ancak, sorguları yanıtlamanın ön koşulu, makinenin belirli bir arama/sorgu için web sayfalarını sınıflandırma yeteneğidir. Web, makinenin web sayfalarını sınıflandıracağı ve ardından belirli bir konu için çıkarımlarda bulunacağı ve içerik üreteceği dev

bir bilgi tabanı olacaktır. O halde, Endüstri, Bilim ve Web'in 4.0 sürümlerinde makineye yeni bir rol atandığı sonucuna varabiliriz: etkin bir eyleyici.

Makinenin etkin bir eyleyici olduğunu, belirli bir bağlamda bir görevi başarmak için amaçlı olarak eylemde bulunduğunda söyleyebiliriz. Endüstri, Bilim ve Web'in 4.0 versiyonlarına geçiş için, kararlar veren, içerik üreten, önerilerde bulunan ve gerçekliğin temsilinin eksikliği ve/veya yanlışlığı nedeniyle belirsizlik altındaki yargılar için açıklamalar sunan, yani otonom ve açıklayabilen bir makineye ihtiyaç vardır. Bu nedenle, makinenin aktif bir eyleyici olması için gerekli olan gereksinimleri tespit etmek ve bunları karşılamak bu alanlardaki gelişmeler için atılacak ilk adımı teşkil eder.

E.4 Anlayabilen Makine

Günümüzde makine, belirli bir görev için belirli bir alanın yapısallaştırılmış ve/veya yapısallaştırılmamış verilerini kullanan istatistiksel modeller ile oluşturulmuş uzman sistemden daha fazla bir şey değildir. Makinenin kararları/tavsiyeleri yalnızca makine için temsil edilen veya eğitilen etki alanında geçerlidir. Yani üretilen sonuçlar başka bir alana uygulanamayabilir ve/veya temsiller başka alanlar için yetersiz olabilir. Ancak Büyük Veri söz konusu olduğunda açık dünya modeline göre üretilmiş sistemler devrede olmalıdır. Akıllı fabrikalardaki makineler, çeşitli ortamlardan kararlar almak zorundadır; Web'deki çeşitli bağlamlardan gelen içerikleri kümelemek zorundadır; farklı amaçlara sahip çeşitli bilimsel çalışmalar arasında otomatik olarak iletişim kurmak zorundadır. Bilgi işlemenin önündeki mevcut en büyük zorluklardan bazıları, çoğu verinin yapısallaştırılmamış olması, anlamsal özelliklerin alan uzmanları tarafından bağlama göre işaretlenmesi, çeşitli akıl yürütme yöntemleri için kullanılabilecek biçimsel sistemlerin çok kısıtlayıcı olmasıdır. O halde, makinenin otonom olabilmesi için veriyi otomatik olarak yapılandırması, anlamsal özellikleri otomatik olarak ilgili bağlama göre tespit edebilmesi ve çeşitli akıl yürütme sistemlerini uygulayabilmesi gerekmektedir. Bunun olabilmesi için ise makinenin *anlayabilir* olması gereklidir.

Makine-anlayabilir bir sistem için makinenin anlamsal özellikleri otomatik olarak işleyebilmesi gereklidir. Makine-okuyabilir sistemlerden alışık olduğumuz üzere entitelerin hangi anlama geldikleri onlara anlamsal özellikler işaretleyerek belirlenir. Ancak burada önemli olan, bağlama göre bu anlamsal özelliklerin tespit edilmesidir. Makine-anlayabilir bir sistem için ise tüm entitelerin tüm bağlamlardaki hallerine göre tüm anlamsal özelliklerinin makinede temsil edilebilmesi gerekir ki bu imkansızdır. Bu nedenle makine-okuyabilir yaklaşımı radikal biçimde değiştirmemiz gerekmektedir.

E.4.1 Entitelerin Temsili

Bir verinin temelde iki bileşeni vardır: değer ve tip. Verinin değeri tipinin çizdiği sınırlar içinde işlenir. Örneğin, karakter tipindeki bir verinin değeri "elma" ve bu verinin tüm karakterlerinin büyük harf ile gösterilmesi isteniyor olsun. Karakter tipi için büyük harf olmak tanımlanmış/meşru bir operasyon olduğundan verinin değeri "ELMA" olarak değişir. Ancak, bu verinin iki katı bulunamaz, çünkü karakter tipi üzerinde tanımlı böyle bir operasyon olamaz. Tüm bunlara ek olarak, makine-okuyabilir sistemlerde veriye bir de anlamsal özellik bileşeni eklenir. Örnekteki veriye "tatlı" özelliğinin eklenmesi ve bu bileşen üzerinden verinin işlem görmesi makinenin verilerin anlamları üzerinden işlem yaptığını gösterir. Makine-okuyabilir sistemlerin oluşturulma nedenleri tam da bağlam içinde entitelerin anlamlarını işleyebilmektir. Ancak, makine-anlayabilir bir sistem oluşturmak söz konusu olduğunda ise tüm bağlamlardaki tüm anlamsal özellikleri işaretlemek imkansızdır. Zira, entiteler anlamlarını ancak ilişkili oldukları entiteler üzerinden kazanırlar. Elmanın tatlı olması bazı bağlamlar için önemliyken, kâğıt ağırlığı olabilmesi gibi bazı bağlamlarda hiçbir anlamı yoktur. Dolayısıyla, makinenin elmanın bir kâğıt ağırlığı olarak da kullanılabileceğini çıkarmasını hedeflemek makine-anlayabilir bir sistem oluşturmanın yaklaşımını vermektedir. O halde, verinin üçüncü bileşeni olan anlamsal özelliklerin tıpkı tipleri gibi önceden tanımlı olarak makinede temsil edilmesi ve makinenin bu anlamsal özelliklerden hangilerini hangi operasyonlarla kullanacağını ilgili bağlama göre belirlemesi gerekmektedir.

E.4.2 Veri

Yukarıda veriden ve verinin anlamsal özellikleriyle gösterilmesi gerektiğinden bahsettik. Bunun nasıl yapılacağına geçmeden önce veriyi ve ontolojiyi incelememizde fayda olacaktır.

Bu çalışmada veriyi makine perspektifinden ele alıyoruz. Yani veri, makinedeki temsilin en temel kavramıdır. Buna göre, Veri 1.0 işlemeye hazır yapısallaştırılmış veri, Veri 2.0 ise makineler arasındaki iletişimi sağlayan veri, Veri 3.0 ise makinenin üst veriler aracılığı ile temsil edilen anlamsal özelliklere sahip verilerdir. Burada dikkat edilmesi gereken birinci nokta Veri 3.0'ın aslında Veri 1.0 da olduğudur. Ancak Veri 2.0 Veri 3.0 olmak zorunda olmamasıdır. İkincisi ise bu üç veri tipinin makine-anlayabilir bir sistemi sağlayamaz olduğudur. Zira makine-anlayabilirliği sağlamak için Veri 4.0'ın inşası gereklidir: Anlamsal özellikleriyle fenomenlerin formel olarak yapısallaştırıldığı veriler.

E.4.3 Ontoloji 4.0

Makine-anlayabilirliği sağlayabilmek için verinin üçüncü bileşeninin anlamsal özellikler olduğunu ancak bunun insanların işaretlemesi ile elde edilemeyeceğini gördük. Zira, anlayabilirlik var olan kavramların yeni bağlamlarda yepyeni kombinasyonlarla kullanılabilmesi becerisidir. Bunu dil üzerinden anlatırsak, bilinen bir dildeki sözcüklerin, dilbilgisi kurallarına uygun olarak farklı bağlamlarda bir araya anlamlı bir şekilde getirilmesidir. Dili konuşan kişi, yeni bağlamlarda belirli sözcükleri bir araya getirerek ifadesini dile getirir. Bu dile getiriş eylemi sonucunda kişinin anlıyor olduğunu düşünürüz. O halde, makine-anlayabilir yapıya geçmenin esas meselesinin entitelerin yeni bağlamlarda nasıl işleneceğini otomatize etmek olduğunu söyleyebiliriz. Ancak, bunun olması için veri yapılandırmamızı değiştirmemiz gerekmektedir.

Bu noktada ontolojilere değinmemiz gerekir. Ontoloji, felsefenin bir alt dalı olarak varlığın ne olduğunu, varlıklar arası ilişkileri inceleyen bir araştırma alanıdır. Metafiziğin alt dalı olarak varlığı inceleyen ontolojilere, bu çalışmada, Ontoloji 1.0 diyoruz. Ontoloji 2.0 ise varlığı formel araçlarla inceleyen ve ontoloji kur-

manın, varlıklardan bahsetmenin ilkelerini formel araştırmalara dayandıran bir felsefe alt dalıdır. Ontoloji 3.0 ise tam da yukarıda incelediğimiz makine-anlayabilir sistemlerin oluşturulması için inşa edilen enformatik sistemlerinde çalışılan bir araştırma alanıdır. Yani, Ontoloji 3.0'da inşa edilen ontolojiler makinede temsil edilmesi istenen alanın varlıklarını ve bu varlıklar arası ilişkilerini formel araçlar ile gösterirler. Bu tür ontolojiler insanların verilere anlamsal özellikleri eklemesi yani verilerin işaret ettikleri entitelere ontolojik statü tayin etmesi yoluyla oluşturulur. Anlaşılacağı gibi de şu ana kadar olan tüm yaklaşımların temelinde nesne ile özelliklerinin ayrı tutulduğu, nesnenin öncelendiği anlayış vardır. Bunun makine-anlayabilirliği açısından işimize yaramayacağı aşikardır. Bu nedenle veri yapılandırmasına yeni bir soluk getirecek yaklaşıma gereksinim vardır.

Felsefi ontolojileri anlamsal özelliklerin temsilinin nasıl olması gerektiği penceresinden yaptığımız araştırma sonrasında *trop kuramı*nın fenomenin temsilinde kullanılması gereken yaklaşım olduğunu savunmaktayız. Makine anlayabilirliği çerçevesinde benimsediğimiz trop kuramına göre var olan her şey, örneğin nesneler, ilişkiler, olaylar, özelliklerin toplamıdır. Böylece nesne ve özellikler arasındaki keskin çizgi ortadan kalkar ve ortada sadece özellikler kalır. O halde, fenomenler veriye yepyeni bir ontoloji içinden dönüşebilirler. Bu tam da makine-anlayabilirliği için makineye özgü bir ontoloji olacaktır. İnsanın dünyayı anlaması ve anlamlandırmasını makineye aktarmak için değil, makinenin dünyayı anlaması için oluşturulmuş bir makine ontolojisi oluşturulacaktır. Biz buna *Ontoloji 4.0* diyoruz.

Trop kuramının metafiziksel tartışmalarını felsefeye teslim ederek, bu kurama bizi yaklaştıran ve bize ilham veren iki görüşten bahsedelim: Wittgenstein'ın *Tractatus Logico-Philosophicus*'u ve Rovelli (2021)'nin *Helgoland*'ı. Bu iki çalışmaya göre nesnelerden bahsedebilmek için ilk önce ilişkilerden bahsetmek gerekir. Başka bir deyişle, nesnelerden ancak ilişkilerin varlığı üzerinden bahsedebiliriz. O halde, her şey ilişkilerle anlatılmalıdır. Bu noktada, makine-anlayabilir bir sistem kurabilmenin temellerini ilişkilere indirgemek doğru olacaktır. Bunlardan

yola çıkarak, entiteleri ilişkiler ile bilebiliyorsak, tropları ilişkiler olarak temsil ettiğimizde entiteleri makinelerde temsil etmiş oluruz. Sonuç olarak, makine-anlayabilirliğinin temelini ilişkilerin troplar olarak temsil edilmesi oluşturur.

E.4.4 Tropların İşlenmesi

Entitelerin tipleri nasıl belirlenecektir? Ya da eğer entiteler tropların bir kompozisyonu olarak temsil edilecekse entiteler üzerine işlemler nasıl yapılacaktır? Entitelerin tiplerinden bahsetmek makine-anlayabilir sistemlere özgü olmalıdır, zira eğer her şey ilişkiler ile anlam kazanacaksa, yani bir varlık, onu diğer varlıklarla ilişkilendiren ilişkilerle ontolojik statüsünü kazanacaksa entitenin makine tarafından işlenebilmesi için kendisinin değil, ilgili bağlamda aktif olan ilişkilerinin işlenmesi gerekecektir. Bu bakımdan, entiteler tiplendirilemezler, ancak troplar ve onların kompozisyonlarının tipleri belirtilir, çünkü ancak o zaman anlamsal özellikler üzerinde işlemler yapılabilir. Özetle, hangi semantik özelliklerin işleneceğini entite değil, o entiteye bağlı ilişkiler belirler; böylece troplar ve trop kompozisyonlarının tiplendirilmesi sayesinde makine-anlayabilir bir sisteme ulaşılabilir. O halde, makine-anlayabilirliğine giden yolda, ilk olarak, troplar tipleştirilmelidir; ikinci olarak, trop ve trop kompozisyonları üzerindeki tip kuralları belirtilmelidir ve son olarak, bu kuralları işletmek için bir biçimsel model tespit edilmedir.

E.4.5 Tip olarak Troplar

Yukarıdaki bölümde gördüğümüz üzere tropların tiplerini ve bu tiplerin kompozisyon kurallarını belirleyebilirsek makineler verileri anlamsal özellikleriyle işleyebilirler. Bununla birlikte makine-anlayabilirliği için tiplerin de tiplerinin makine tarafından belirlenmesi gereklidir. Aksi takdirde makine-okuyabilirliğin ötesinde bir temsilden bahsediyor olmayız. Ayrıca tiplerin tiplerinin ve hatta tiplerin tiplerinin tiplerinin makine tarafından tanımlanabiliyor olması arka plan bilgisinin makinede temsil edilebildiğini de gösterir. Örneğin, "Tüm canlılar ölümlüdür" ve "Ali öğrencidir" cümlelerinden yola çıkarsak, "öğrenci"nin anlamsal özelliklerinden birinin "insan olmak," "insan" olmanın anlamsal özelliklerinden birinin "canlı olmak" olduğunu bildiğimizden, Ali'nin de ölümlü olduğu

sonucuna varabiliriz. İşte, 'canlı,' 'insan,' 'ölümlü,' 'Ali' ve 'öğrenci' entitelerinin anlamsal özellikleri arasında makinenin de aynı şekilde işlem yapabilmesi gerekir. Ancak, tiplerin tiplerinden bahsedebilmek için ele alınan tip kuramının bu tarz bir işleme izin verebiliyor olması gerekir, aksi takdirde küme kuramı temelli tip kuramlarındaki en büyük sorun olan kendisi tip olmayan ama tüm tipleri içeren şeyin de tipinden bahsetmek zorunludur ama makine de bunu hesaplayamaz. Bu soruna çözüm ise Martin-Löf'ün tip kuramıdır. Bu kurama göre tiplerin tiplerinden yapısalcı perspektiften yola çıkarak, tüm tipleri içeren bir tipten, sonra bu tipi içeren bir üst tipten oluşan bir hiyerarşiden bahsedilebilir. Böylece, yeni tiplerin oluşturulabilmesi ve tiplerin tiplerinden bahsedilmesi mümkün olur.

Entiteleri troplar ile temsil ederken izleyeceğimiz yolu bulduğumuza göre Martin-Löf'ün iddia ettiği en küçük olan tiplerin kümesini tespit etmeye başlayabiliriz. Bu şu demek: Öyle bir troplar derlemesi bulalım ki bunları kullanarak diğer troplar, ve aynı şekilde diğer tipler de türetilebilsin. Ancak bu anlamsızdır: Ne insanlar ne de makineler böyle bir derlemeden söz edebilir. Aksine, alt tiplerin de üst tipler gibi makine tarafından sürekli türetilebiliyor olması gerekir. Bunlar makineye hazır bir biçimde sunulmadığı takdirde makinenin bunları belirlemesinin bir yolu yoktur. Ayrıca, uzamsal olan her şey sadece belirli bağlamlarda belirli rolleri alır, belirli davranışlarda bulunur. Burada makine-anlayabilir sistemlerde karşılaştığımız bağlama göre nesnelerin anlamsal özelliklerinin tespitinin insan gücüne bağlı olma sorunu ile karşı karşıyayız. O halde, tropların tiplerinin de bağlama göre makine tarafından otomatik olarak belirlenmesi gerekir. O halde, tropların uzamsal olması makine-anlayabilirliğinin önünde bir engeldir. Bu sorunu çözmek için bilgisayar bilimine danışacağız.

E.5 Urtrop Kuramı

Tropların tipleştirilmesi onların işlenebilmesi için elzemdir. Dolayısıyla, temsil anlayışımız olan tropların ve onların komposizyonlarının entiteleri oluşturmasından ya da anlamsal özelliklerin işlenmesi için kullanmamız gereken Martin-Löf'ün tip kuramından vazgeçemeyiz. Ancak, tropların uzamsal olmaları onların makinede temsilini imkansızlaştırır. Cardelli ve Wegner'e (1985) göre tipleştirdiğimiz anda entiteye bir elbise biçiyor ve onun rolünü de anında be-

lirlemiş oluyoruz. Yani, troplar *kıyafetli* temsillerdir ve bu tür elbiseler hangi tropun hangi troplar ile etkileşime gireceğini belirler. Ancak temsiller çıplak olursa tropların tiplerinden kendi başlarına bahsedemeyiz. Bu fikir de bizi *tip bağımsız özellik kuramları*na götürür.

Tip bağımsız özellik kuramlarına göre tipler tipsiz unsurların bir araya gelmesiyle oluşur. Diğer bir deyişle, tipsizlerin bir araya gelmesi bir rolü ya da bir davranışı belirler. Bu kuramların kullanımının bizim açımızdan en önemli özelliği tipsiz unsurların sayısız bir araya gelişlerinden tipler oluşturulabilmesi ve tipsiz bir unsurlar aracılığı ile tipler tanımlandığı için, bir tipi tanımlamak için başka bir tipe başvurmaya gerek kalmayacağıdır. Bununla da hesaplanabilirlik sorununun önüne geçilmiş olur. Diğer yandan, tipsiz unsurları birbirleri ile tanımlamada bir sıkıntı olmayacaktır, çünkü bunlar uzamsal olmadıkları için döngüsel yapıların var olması kuramı geçersiz kılmaz. O halde, makine-anlayabilirliği yolunda verinin troplar ile değil, uzamsal olmayan ve kompozisyonları tropları verecek ontolojik unsurlar ile temsil edilmesi gerekir. Almanca'da öncel, ilk, temel anlamlarına gelen *Ur-* önekini 'trop' kavramına ekleyerek oluşturduğumuz *urtrop* kavramı ile makine ontolojisindeki tüm entiteler oluşturulabilir.

Urtrop kuramına göre, urtroplar uzamsal değildir. Uzamsal olmamak demek hiçbir anlamı olmaması da demektir. Tüm troplar bir ilişkiyi işaret ettikleri için hepsinin belirli bağlamlarda anlamları vardır. Ancak urtroplar başka urtroplar ile bir araya gelse bile bir tropa işaret ederler; kendilerinin bir anlamı yoktur. Yani, onların kompozisyonları tek başına hiçbir anlamı olmayan uzamsal şeylere yol açar, zira tüm troplar anlamlarını ancak başka troplar ile ilişkilendiğinde kazanır. Bununla birlikte, bir urtrop kompozisyonun, bir özne veya bir bağlamdaki bir ilişki gibi statik rolleri yoktur; bir kompozisyon bunlardan herhangi biri olabilir. Bu nedenle, Cardelli ve Wegner'in (1985) açıkça belirttiği gibi, tipler tipsiz evrenlerden başlayarak doğal olarak ortaya çıktığından urtrop kuramına dayalı bir makine ontolojisi inşa etmek kesinlikle meşrudur.

E.6 Urtrop Kuramının Biçimselleştirilmesi

Şu ana kadar tropların tip kuramına göre işleneceğini, makine ontolojisinin uzamsal olmayan şeyler üzerine kurulacağını ve tropların da bu şeylerin kompozisyonu olarak temsil edileceğini gördük. Kısaca, bir makine ontolojisini oluşturmak için ontolojik ve berimsel temel olarak urtrop kuramını sunduk. Peki, urtrop kuramına en uygun biçimsel sistem ne olmalıdır?

E.6.1 Kategori Kuramı

Araştırmalarımız tip bağımsız sistemlerin temsilleri için farklı biçimselleştirme araçları kullanılabileceğini gösterse de *kategori kuramı*nın makine ontolojisi kurmak için en uygun biçimselleştirme aracı olduğu sonucuna vardık. Bunun nedeni kategori kuramının matematiğin temeli olarak kabul edilecek kadar yüksek bir soyutlama düzeyi sağlaması değil, urtrop kuramı ile ortak özellikler taşımasıdır. Kategori kuramındaki nesneler aynen urtrop kuramında olduğu gibi içeriksiz ve uzamsızdır. Yine, kategori kuramının da ilişki temelli bir yaklaşımı vardır: Tüm hesaplamalar ilişkiler üzerinden gerçekleşir. Tüm bunlara rağmen, kategori kuramını doğrudan alıp makine ontolojisini kurmak için kullanmak doğru bir tutum olmaz. Bu nedenle, urtrop kuramını formel düzeyde temsil etmek için bazı aksiyomlar ve kurallarla kuramın sınırları belirlenmelidir. Örneğin, tip oluşturma veya eşzamanlılık kuralları, kategori kuramına ek ilkeler gerektirir. Böylece, kategori kuramı bu tür eklemelerle urtrop kuramını biçimselleştirebilir.

Urtrop kuramındaki nesnelerin kategori kuramında nasıl temsil edileceği noktasında *Curry–Howard–Lambek* uyuşmasına başvururuz. Buna dayanarak, Peruzzi (2006), Kartezyen kapalı kategoriler ile tipli lambda kalkülüsün, terminal nesnesi olmayan C-monoidler ile tipsiz lambda kalkülüsün, lokal Kartezyen kapalı kategoriler ile Martin-Löf tip kuramlarının ve toposlar ile sezgisel (lokal) tip kuramlarının uyuştuğunu gösterir. Buna dayanarak, urtropları terminal nesnesi olmayan C-monoidlerle temsil edilebileceğini iddia ediyoruz. C-monoidlerin nesne olduğu Kartezyen kapalı kategorilerle de tropları ve bu kategorilerin nesne olduğu başka kategoriler kurarak da entitelerin makinelerde temsil edilebileceğine kolaylıkla erişebiliriz. Ancak, özel bir Kartezyen kapalı kategori olan

toposların tropların temsili için çok daha doğru bir biçimsellik taşıdığını savunuyoruz. Çünkü, matematikte toposlar sayesinde bir 'köprü' topos oluşturularak iki farklı matematiksel kuram incelenebilir, karşılaştırılabilir ve analiz edilebilir; alt nesne sınıflandırıcı nesnesi sayesinde toposların mantıksal yönleri keşfedilebilir ve hatta oluşturulabilir. Bu nedenle topos kuramsal yöntemler, matematiksel kuramlar arasında derin bağlantıların bulunmasını sağlar ve bu bağlantılar aracılığıyla bilgi transferleri mümkündür (Caramello, t.y.). Toposların bu özelliklerini troplara kazandırmak temsil sisteminin gücünü arttırmaktır. Çünkü, doğruluk değerlerini troplar içinde ve arasında incelemek, tropların neliği hakkında araçlar elde edebilmek makine-anlayabilirliği için gereklidir. Bu yüzdendir ki topos kullanımı hem ontoloji hem de veri bilimi çalışmalarında dikkat çekmiştir. Spivak (2015), Ulusal Standartlar ve Teknoloji Enstitüsü'nün kategori kuramının Büyük Veri çağında potansiyel bir matematiksel temel olarak tanıdığını bildirmektedir ve kendisinin de üyesi olduğu Topos Enstitüsü'nde Büyük Veri'nin işlenmesi ile ilgili kategori kuramı temelli projeler yürütülmektedir.

E.7 Sonuç: Bir Makine Ontolojisi olarak Ontoloji 4.0

Bu çalışmanın amacı makine-anlayabilirliğinin ilkelerini tespit etmekti. Araştırmamız yeni bir veri yapılandırmasına gitmemiz gerektiğini gösterdi. Bu doğrultuda, anlamsal özelliklerin, değer ve tip gibi verinin bir bileşeni olarak temsil edilmesinin makine-anlayabilirliğini sağlayacağını iddia etmiş, bu yüzden de entitelerin makinede temsilleri için trop kuramsal bir yaklaşım gerektiğini savunmuştuk. Böylece, makine-anlayabilir sistemlerin sıklıkla kullandığı ontolojik yaklaşım (Ontoloji 3.0) yerine, fenomenin ontoloji içinde temsili olan Ontoloji 4.0'ın makine-anlayabilir verinin (Veri 4.0'ın) temeli olacağını ifade etmiştik. Ontoloji 4.0'ın işlenebilmesi için Martin-Löf tip kuramından faydalanmak gerektiğini, ancak tropların uzamsal olmalarından dolayı berimsel yapıda karşılaşılan sorunları çözmek için tip bağımsız sistemlerin ontolojik yapıyı oluşturması gerektiği sonucuna ulaştık. Ortaya attığımız urtrop kuramı ile makine ontolojisindeki her şeyin en temelde tipsiz unsurların kompozisyonlarından türediğini, daha açık söylersek, urtrop kompozisyonlarının tropları, trop kompozisyonlarının entitileri,

entite kompozisyonlarının ise daha büyük yapıları var ettiğini gördük. En son olarak da urtrop kuramını makinede uygulayabilmek için kategori kuramından faydalanacağımızı anlattık. Kategori kuramının seçim nedeni ise urtrop kuramı gibi nesnelerin içeriksiz olmaları ve hesaplamaların ilişkiler üzerinden yapılmasıydı. Kategori kuramının tip kuramları ile uyumlanmasını kullanarak da urtropları C-monoidlerle, geri kalan tüm yapıların ise toposlarla temsil edilebileceğini savunduk. Böylece, her şey –yani bir urtrop ile bir entite, bir ilişki ile başka bir ilişki ya da bir bağlam ile bir nesne– arasında ilişkilendirmelerin yapılandırılmasına olanağın olduğunu anlattık. Araştırmamızın sonucu olarak Ontoloji 4.0'ın makine-anlayabilirliğini sağlayacağını göstermiş olduk.

Her ne kadar araştırmamızı tamamlamış ve hedefimize ulaşmış olsak da Ontoloji 4.0'ın özelliklerini inceleyerek sonuçlarımızın sağlamasını da yapmak isteriz. Bunun için ilk önce Ontoloji 4.0'ın işleyeceği çevre olan Endüstri, Bilim ve Web'in 4.0 versiyonlarının özelliklerini özetlemek gerekir.

Her üç alan da birer açık sistemdir, yani sistem içindeki entiteler sistem dışındaki diğer entiteler ile de etkileşim halindedir ve bu etkileşimleri sistem içindeki rollerini ve davranışlarını etkiler. Örneğin, akıllı fabrikalar açık sistemlerdir, çünkü makine çeşitli kaynaklardan gelen verileri gerçek zamanlı olarak manipüle eder. Bu üç alanın ikinci özelliği dinamik olması ve evrimleşmesidir. Etkileşimli oldukları için her etkileşim dinamik bir ortam yaratır; her etkileşimde bu sistemler değişir. Etkileşimler doğrusal olmak zorunda değildir, eşzamanlı da olabilirler. Bu nedenle, bu alanların 4.0 versiyonları dinamikliği ve evrimselleşmesi, eşzamanlılığın temsili ile doğrudan ilişkilidir. Bahsetmek istediğimiz son özellik, etkileşim ve tabii ki berimsel açısından karmaşık sistemler olmasıdır. Etkileşim ne kadar fazlaysa, etkileşimli karmaşıklık da o kadar fazladır (Wegner, 1998). Özetle, Ontoloji 4.0'ın varlığını sürdürdüğü ortamın özellikleri açık, eksik, koşullu, dinamik ve karmaşık olarak özetlenebilir. Şimdi Ontoloji 4.0'ın özelliklerini inceleyelim ve bu ortamın sorunlarıyla ilgilenenleri analiz edelim.

E.7.1 Ontology 4.0'ın Özellikleri

Temsil Standardı: Uzamsal olmayan urtroplar temsillerin standardizasyonunu sağlar: nesneler, ilişkiler, süreçler, kısaca tüm entiteler tek bir düzleme, urtroplara indirgenebilir. O halde, Ontoloji 4.0 bu tipsiz yapı taşları sayesinde makinedeki gösterimin standardizasyonunu garanti eder.

Sözdizimi ve anlambilimin birleşimi: Topos kuramı sembolik ifadeleri ve çıkarımlarını eşitler. Dolayısıyla, bir ispat hem kendi makinedeki ontolojik statüsüne ve kendine içkin olan anlamsal değere sahiptir. Sonuç olarak, topos kuramsal yapılarının tipleri adlandırdığı kategori kuramsal sistemin ontolojik yapı taşları olan urtroplar sayesinde, Ontoloji 4.0 sözdizimi ve anlambilimin bir birleşimidir.

Süreç tabanlı bir çerçeve: Makine-anlayabilirliğinin temelinde her şeyin ilişkiler içinde anlam kazandığı görüşü vardır. Bu doğrultuda da Ontoloji 4.0, her şeyin ilişkilerle temsil edildiği bir çerçeve sunacak şekilde inşa edilmiştir.

Dinamik ve evrimleşen sistem: Urtropların sayısız kompozisyonundan bahsedebildiğimiz için entitelerin bağlam içinde sabit rollere bürünmelerine gerek yoktur. Yani, Ontoloji 4.0'ın dinamikliği urtropların yeniden düzenlenebilir olmasından kaynaklanır: farklı bileşimleri, farklı bağlamlarda ve tiplerde farklı anlamsal özelliklere yol açar. Bu nedenle, urtrop ve troplar gibi Ontoloji 4.0 da dinamiktir. Aynı zamanda her yeni bağlamsal seviye analizinin yeni anlamsal özelliklerin entitelere dahil edilmesi için yeni trop kompozisyonları sağlayabileceği için Ontoloji 4.0'ın evrimleştiğini söyleyebiliriz. Başka bir deyişle, Ontoloji 4.0, entitelere yeni trop kompozisyonlarını temsil edebilen bir çerçeve sağlar.

Eşzamanlılık ve etkileşim kuramı: Kategori ve urtrop kuramlarının Ontoloji 4.0'ı oluşturması, eşzamanlılık ve etkileşimin ilkelerinin gerçekleşmesi için gerekli ve yeterlidir.

Açık bir sistem: Her ne kadar açık sistemlerle başa çıkmak için oluşturulsa da Ontoloji 4.0'ın kendisi de açık bir sistemdir, zira tüm bileşenleri

ve onlardan türeyen her şey etkileşim içindedir.

Birlikte çalışabilir bir sistem: Birlikte çalışabilirlik temsilde standardizasyon gerektirir. Ontoloji 4.0 bunu iki açıdan sağlar: Urtrop kuramı sayesinde entiteleri ontolojik düzeyde ve kategori kuramı sayesinde mantıksal yapıları standartlaştırır.

Kendi kendini organize eden bir sistem: Bir sistem, kendi iç süreçleri tarafından yapılandırıldığında kendi kendini organize eden olarak adlandırılır; yani dış kontrol olmaksızın yapısını oluşturur. Urtrop kuramı sayesinde Ontoloji 4.0 kendi iç süreçleriyle kendini yapılandırabilir.

Üretken bir sistem: Urtrop kuramı sayesinde, makine, bir bağlama göre otomatik olarak yeni troplar, yeni trop kompozisyonları, tiplerin yeni tiplerini ve yeni ontolojilere kadar her şeyi üretebilir.

Sistematik bir sistem: Ontoloji 4.0 üretkendir; ancak, üretilenler tek kullanımlık değildir. Örneğin, yeni bir tip tespit edildiğinde, farklı bağlamlarda kullanılmak üzere saklanır. Bu açıdan Ontoloji 4.0, üretilenleri diğer ilgili yapılara uygulama yeteneğine sahip olduğundan sistematiktir.

Verimli bir sistem: Verimlilik, çok sayıda parçadan sonsuz sayıda bütün yaratma potansiyeline sahip olmakla ilgilidir. Bu tanım ışığında Ontoloji 4.0'ın üretken olduğunu iddia etmek meşrudur çünkü makine-anlayabilirliği tam da entitelerin anlamsal özellikleri üzerinden onları farklı bağlamlarda kullanabilme becerisini içerir. Bunu da urtrop ve trop kuramlarına dayandığı için yapabilir.

İlişkisel bir sistem: Anlamsal özellikleri ve entiteler, ilişkiler, bağlamlar ve ilişkiler, ilişkiler ve olgular, vb. arasındaki morfizmleri ilişkilendirme yeteneği, Ontoloji 4.0'ı ilişkisellikleri bulan en güçlü araç haline getirir.

Özgönderimsel bir sistem: Urtrop kuramı sayesinde, Ontoloji 4.0, otomatik olarak bağlama özel oluşturduğu ontolojileri temsil eden aynı biçimsel yapı ile kendini de temsil edebildiğinden kendi kendine referanslıdır.

Sorguya hazır yapı sağlayıcı: Ontoloji 4.0 herhangi bir veri yığınını otomatik olarak yapısallaştırır ve bu ontolojilerde işlem yapabilmek için başka bir biçimsel yapıya ya da mantık sistemine ihtiyacı da yoktur.

Otopoietik bir sistem: Ontoloji 4.0'ın birliği kategori kuramsal yapılar, tipleme kuralları ve yeni bileşenler oluşturan urtrop kompozisyonlarının organizasyonudur. Bu birlik, ontolojideki her bir bileşenin rolünü/davranışını belirlediği için de Ontoloji 4.0 otopoietiktir.

Ontolojilerin Turing makinesi: Nasıl ki hesaplanabilen her yapı Turing makinesinde temsil edilebiliyorsa, her bağlama ait ontolojiler de otomatik olarak Ontoloji 4.0 tarafından oluşturulur.

Otonom/Etkin eyleyici: Bu bölümde saydığımız özellikler tezin sonucu, bu özellik ise Ontoloji 4.0'ın tüm özelliklerinin bir sonucudur. Ontoloji 4.0, belirli bir amaç/durum için belirli bir bağlamı yapılandırıp, analiz ederek ve yorumlayarak ve ardından bu bağlam hakkında çıkarımlarda bulunup karar verebildiği için otonomdur, /etkin bir eyleyicidir.

Yukarıda Ontoloji 4.0'ın hangi özelliklere sahip olduğuna kısaca değindik. En son vardığımız özellik olan Ontoloji 4.0'ın otonom bir yapıya sahip olması da tezimizin hedefine ulaştığının kanıtıdır.

F. THESIS PERMISSION FORM/TEZ İZİN FORMU

ENSTİTÜ / INSTITUTE

Fen Bilimleri Enstitüsü / Graduate School of Natural and Applied Sciences ☐
Sosyal Bilimler Enstitüsü / Graduate School of Social Sciences ☒
Uygulamalı Matematik Enstitüsü / Graduate School of Applied Mathematics ☐
Enformatik Enstitüsü / Graduate School of Informatics ☐
Deniz Bilimleri Enstitüsü / Graduate School of Marine Sciences ☐

YAZARIN / AUTHOR

Soyadı / Surname : Yargan
Adı / Name : Dilek
Bölümü / Department : Felsefe / Philosophy

TEZİN ADI / TITLE OF THE THESIS (**İngilizce** / English): A Computational Ontological Model For Machine-Understandable Data In Artificial Intelligence

TEZİN TÜRÜ / DEGREE: **Yüksek Lisans** / Master ☐ Doktora / PhD ☒

1. **Tezin tamamı dünya çapında erişime açılacaktır.** / Release the entire work immediately for access worldwide. ☒
2. **Tez iki yıl süreyle erişime kapalı olacaktır.** / Secure the entire work for patent and/or proprietary purposes for a period of **two years**.* ☐
3. **Tez altı ay süreyle erişime kapalı olacaktır.** / Secure the entire work for period of **six months**.* ☐

** Enstitü Yönetim Kurulu kararının basılı kopyası tezle birlikte kütüphaneye teslim edilecektir. / A copy of the decision of the Institute Administrative Committee will be delivered to the library together with the printed thesis.*

Yazarın imzası / Signature ..... **Tarih** / Date .....

Tezin son sayfasıdır. / This is the last page of the thesis/dissertation.