CROSS-LEVEL TYPING THE LOGICAL FORM FOR OPEN-DOMAIN SEMANTIC PARSING

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

BY

İSMET ADNAN ÖZTÜREL

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

AUGUST 2022

Approval of the thesis:

CROSS-LEVEL TYPING THE LOGICAL FORM FOR OPEN-DOMAIN SEMANTIC PARSING

submitted by **İSMET ADNAN ÖZTÜREL** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Cognitive Science Department, Middle East Technical University** by,

Prof. Dr. Deniz Zeyrek Bozşahin
Director, **Graduate School of Informatics** _____

Dr. Ceyhan Temürcü
Head of Department, **Cognitive Science** _____

Prof. Dr. Cem Bozşahin
Supervisor, **Cognitive Science, METU** _____

Examining Committee Members:

Prof. Dr. Halit Oğuztüzün
Computer Engineering, METU _____

Prof. Dr. Cem Bozşahin
Cognitive Science, METU _____

Assoc. Prof. Dr. Aysun Kunduracı
Foreign Language Education, Yeditepe University _____

Assist. Prof. Dr. Umut Özge
Cognitive Science, METU _____

Assist. Prof. Dr. Murat Ulubay
Business Administration, Ankara Yıldırım Beyazıt University _____

Date: 29.08.2022

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, Surname: İsmet Adnan Öztürel

Signature :

ABSTRACT

CROSS-LEVEL TYPING THE LOGICAL FORM FOR OPEN-DOMAIN SEMANTIC PARSING

Öztürel, İsmet Adnan
Ph.D., Department of Cognitive Science
Supervisor: Prof. Dr. Cem Bozşahin

August 2022, 116 pages

This thesis presents a novel approach to assigning types to expressive Discourse Representation Structure (DRS) meaning representations. In terms of linguistic analysis, our typing methodology couples together the representation of phenomena at the same level of analysis that was traditionally considered to belong to distinctive layers. In the thesis, we claim that the realisation of sub-lexical, lexical, sentence and discourse-level phenomena (such as tense, word sense, named entity class, thematic role, and rhetorical structure) on the surface can be represented as variations of values that belong to the same typed category within our cross-level typing technique.

We show the implications of our approach on the computational modelling of natural language understanding (NLU) using Combinatory Categorial Grammar, specifically in the context of one of the core NLU tasks, semantic parsing. We present that cross-level type-assigned logical forms deliver compact lexicon representations and help re-formalise search space constraining tasks, such as Supertagging, as part of the semantic analysis, whereas such approaches were only used in syntactic parsing. We empirically demonstrate the effectiveness of using a training objective that is based on masking the typed logical forms in pre-training models to obtain re-usable lexical representations. Our results indicate that improved model performance on parsing open-domain text to DRS is possible when the embedding layer of an encoder-decoder model such as Transformer is initialised with weights that are distilled from a model that is pre-trained using our objective.

Keywords: Semantic Parsing, Typed Logical Form, Discourse Representation Theory, Combinatory Categorial Grammar, Pre-Trained Embeddings

ÖZ

AÇIK ALAN ANLAMBİLİMSEL AYRIŞTIRMA İÇİN MANTIKSAL FORMA DÜZEYLER ARASI TÜR ATANMASI

Öztürel, İsmet Adnan
Doktora, Bilişsel Bilimler Bölümü
Tez Yöneticisi: Prof. Dr. Cem Bozşahin

Ağustos 2022 , 116 sayfa

Bu tez Söylem Gösterim Yapısı (SGY) anlam temsillerine tür atamak için yeni bir yaklaşım sunmaktadır. Dilbilimsel analiz açısından, kullandığımız tür atama yöntemi, geleneksel olarak farklı katmanlara ait olduğu düşünülen görüngülerin temsilini aynı analiz düzeyinde bir araya getirmektedir. Tezde, sözcük altı, sözcük, tümce ve söylem düzeyindeki olguların (zaman kipi, sözcük anlamı, adlandırılmış varlık sınıfı, tematik rol ve retorik yapı gibi) yüzey oluşumlarının aynı tür kategoriye ait değerlerin varyasyonları olarak temsil edilebileceğini ileri sürmekteyiz.

Yaklaşımımızın Doğal Dil Anlama görevlerinden biri olan anlamsal ayrıştırma bağlamında, Birleşenli Ulamsal Dilbilgisi kullanarak, hesaplamalı modellemeye ilişkin etkilerini göstermekteyiz. Düzeyler arası tür atanmış mantıksal formların, sıkıştırılmış sözlük temsilleri sağladığını ve yalnızca sözdizimsel ayrıştırmada kullanılan Süper Etiketleme gibi arama alanını kısıtlayan görevlerin anlamsal analizin bir parçası olarak biçimlendirilmesine yardımcı olduğunu sunmaktayız. Ayrıca, önceden eğitilen modellerde mantıksal formların maskelenmesine dayanan bir hedef kullanmanın yeniden kullanılabilir sözcük temsilleri elde etmedeki etkinliğini göstermekteyiz. Sonuçlarımız, Transformer gibi bir kodlayıcı-kod çözücü modelin gömme katmanının sunduğumuz hedef kullanılarak önceden eğitilmiş bir modelden damıtılmış ağırlıklarla başlatılmasının, açık alan metinlerinin SGY'ye ayrıştırılmasında model performansını arttırdığını göstermektedir.

Anahtar Kelimeler: Anlamsal Ayrıştırma, Tür Atanmış Mantıksal Form, Söylem Gös-

terim Kuramı, Birleşenli Ulamsal Dilbilgisi, Ön Eğitimli Vektörler

To my beloved family

Pınar, Lale & Necil

ACKNOWLEDGMENTS

Completing this thesis would be difficult without the support of many people, whom I owe a debt of gratitude here.

I would like to thank my supervisor, Cem Bozşahin, for the insightful perspectives and invaluable input that he provided to this thesis. He was the one who first introduced me to research just after finishing my undergraduate studies, and I'm deeply grateful for his continuous guidance, support, and friendship whenever I encountered difficulties navigating the research landscape.

I would like to thank Halit Oğuztüzün, Umut Özge, and Cengiz Acartürk for their guidance throughout the thesis committee meetings. My additional committee members, Aysun Kunduracı and Murat Ulubay, provided valuable input, which helped me shape this thesis to its final form. Their comments and criticism substantially improved the quality of this research, which can be traced in many aspects throughout this manuscript. I was first introduced to Discourse Representation Theory by Ceyhan Temürcü. I'm thankful to him for the foundation that he provided me on theories of meaning.

I would also like to thank Alexander Gutkin, Slav Petrov, Tom Kwiatkowski, Michael Collins, and Ryan McDonald for reviewing this thesis or its earlier derivatives. Their valuable comments significantly helped me keep on track and shaped my research direction. I'm grateful to Elif Selvi Gök for all the engaging discussions and enduring all my questions and to Tolga Kayadelen for kindly helping me with the intricacies of linguistic theory. I'm thankful to Clara Rivera for her efforts to leverage my outreach and to Sibel Gülnar for helping me with the administrative processes and for being a wonderful colleague throughout my time at METU. My friends and colleagues, Murat Torun, Hasan Gökhan Tezcan, Kerem Eryılmaz, and Işın Demirşahin kept me motivated all through my PhD.

Lastly, it is hard to put into words the sustained support that Nermin Pınar Erdoğan Öztürel, Lale Öztürel, and Necil Öztürel gave to me. They were always available to bounce ideas off, to keep me going, and to take care of me in any way they could. Without them, not only this thesis but life would be incomplete.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

:=	Type assignment operator in CCG
:	Syntatic and semantic type correspondence in CCG, or DRS label assignment in DRT
/	Forward slash in CCG
\	Backward slash in CCG
|	Slash with under-specified directionality in CCG
$\Rightarrow$	Codomain of a function, or material implication
$>$	Forward function application in CCG
$<$	Backward function application in CCG
$>\mathbf{B}$	Forward harmonic composition in CCG
$<\mathbf{B}$	Backward harmonic composition in CCG
$>\mathbf{B}_{\times}$	Forward crossing-composition in CCG
$<\mathbf{B}_{\times}$	Backward crossing-composition in CCG
$>\mathbf{B}^{n}$	Forward generalized composition in CCG
$<\mathbf{B}^{n}$	Backward generalized composition in CCG
$>\mathbf{T}$	Forward type-raising in CCG
$<\mathbf{T}$	Backward type-raising in CCG
′	Logical constant specifier
3s	Third-person Singular
$\mathbf{B}$	Composition combinator in CCG
AMR	Abstract Meaning Representations
BPE	Byte-Pair Encoding
CG	Categorial Grammar
CCG	Combinatory Categorial Grammar

CFG	Context-Free Grammar
CFPSG	Context-Free Phrase-Structure Grammar
CRF	Conditional Random Field
DRS	Discourse Representation Structure
DRT	Discourse Representation Theory
EPDA	Embedded Pushdown Automata
FOPL	First-Order Predicate Logic
GAT	Graph Attention Network
GBT	Government and Binding Theory
GCN	Graph Convolutional Network
GMB	Groningen Meaning Bank
HG	Head Grammar
HMM	Hidden Markov Model
HPSG	Head-Driven Phrase Structure Grammar
IPA	International Phonetic Alphabet
L-DRT	Layered Discourse Representation Theory
LF	Logical Form
LFG	Lexical Functional Grammar
LIG	Linear-Indexed Grammar
LSTM	Long Short-term Memory
LTAG	Lexicalized Tree-Adjoining Grammar
MDC	Maximise Discourse Coherence
MLM	Masked Language Modelling
P-DRS	Projective Discourse Representation Structure
P-DRT	Projective Discourse Representation Theory
PCCG	Probabilistic Combinatory Categorial Grammar
PDA	Pushdown Automaton

PMB	Parallel Meaning Bank
POS	Part-of-speech
PTB	Penn Treebank
RNN	Recurrent Neural Network
S	Substitution combinator in CCG
S-DRS	Segmented Discourse Representation Structure
S-DRT	Segmented Discourse Representation Theory
T	Type-raising rule in CCG
TAG	Tree-Adjoining Grammar
WSJ	Wall Street Journal Corpus

CHAPTER 1

INTRODUCTION

Compositionality is at the core of linguistic analysis, dating back to Fregean semantics. The Montague grammar (Montague & Thomason, 1975) explicitly formalises an interface between the syntactic categorisation of the constituents and their corresponding meaning as part of the lexicon. In this framework, the lexical meaning is usually represented with first-order predicate logic (FOPL) that is abstracted using lambda calculus. The process of building the interpretation for a complete sentence involves combining the meaning of parts using the layer of abstraction that lambda calculus provides over the order that syntax dictates. A similar notion of compositionality is also present in lexicalised grammar theories, such as Combinatory Categorial Grammar (CCG; Steedman, 1996, 2000) or Tree-Adjoining Grammar (TAG; Joshi and Schabes, 1997), where complex descriptions of local syntactic and semantics constraints for the constituents are defined as part of the lexicon, while they differ from each other on how they encode syntactic sub-categorisation, combine constituents, and capture unbounded constructions.

This thesis is concerned with building compositional interpretations that are expressive enough to uniformly represent the phenomena that are observed in open-domain meaning and that are traditionally analysed as part of disjoint semantic analysis levels. We work with a lexicalised grammar theory, CCG, because it provides a clear syntax-semantics interface at the lexical-level. Therefore, we assume that any local semantic constraints that we introduce as part of the lexical semantic type will also manifest themselves in higher-level interpretation after it is compositionally derived.

To represent meaning, we recruit a dynamic semantics theory, Discourse Representation Theory (DRT; Kamp, 1981; Kamp and Reyle, 1993). This is because when its segmented (Asher, 1993; Asher & Lascarides, 2003; Lascarides & Asher, 2008) and projective (Venhuizen, Bos, et al., 2013) extensions are paired with neo-Davidsonian representation of event semantics, the minimal meaning-bearing unit of DRT, which is Discourse Representation Structures (DRS), captures a range of phenomena including tense, named entity class, word sense, thematic role, and discourse relations over decoupled sets of discourse referents and conditions that represent predicate-argument structure. DRS language can be defined as a first-class citizen of a lambda calculus language to introduce the compositional interface. Hence, it is an adequate representation which can replace comparably shallow FOPL meaning representations that are inherited from the Montague grammar and that do not define a principled method of uniformly capturing phenomena beyond the scope of a sentence.

In terms of linguistic analysis, the main contribution of this thesis is to introduce a principled methodology to couple the representation of previously mentioned cross-level phenomena as functionally equivalent values with respect to a structural description of the meaning representation. That is achieved by assigning types to the meaning representation in hand. The thesis introduces a type assignment methodology on DRS meaning representations that is solely dependent on the syntax of the logical form. The approach we take entails that representation of cross-level phenomena belongs to the logical form and, therefore, should be specified as part of the lexical semantic type but not the syntactic type. The neutrality of the structure of the logical form language, as an intermediate representation of meaning, to the syntactic properties of the language that is being analysed, ensures the universal applicability of the proposed type assignment methodology across typologically different languages.

In this context, we first present a grammar that describes the syntax of DRS logical forms. This grammar acts as the sole source for inducing a type inventory. Then, we consider DRS meaning representations as expressions of this well-defined logical form language and assign types to the constituents that make up a DRS. When DRS logical forms are typed using this self-contained type assignment methodology, we observe that representation of sub-lexical (such as tense), lexical (such as named entity class or word sense), sentence-level (such as thematic role), and discourse-level phenomena (such as rhetorical relations), are coupled under the same type. For instance, coupling cross-level phenomena as part of the logical form over a type system in this manner alleviates the necessity to introduce discourse-level descriptions of rhetorical structure that take sentences as arguments of discourse predicates (Mann & Thompson, 1987; Webber, 2004). Discourse relations, just like tense or thematic role, can be represented as instantiations of a value as part of the lexical semantic type, which is propagated beyond the sentence scope through compositionally built interpretation.

Type assignment on the logical form also has implications in terms of computational modelling. The focus of this thesis, from a modelling perspective, is on semantic parsing. That is the task of mapping natural language to the logical form. We can point out two lines of research within this problem scope that are relevant to the motivation of this thesis.

The first one is sequence-to-sequence models, together with transfer learning via pre-training generalised models and then fine-tuning them on downstream tasks, which has resulted in a recent paradigm shift in computational modelling (Devlin et al., 2019; Howard & Ruder, 2018; Peters et al., 2018; Radford et al., 2019). Semantic parsing is no exception to this trend. In particular, the approach formalises the problem as a translation task where sentences from a source natural language are mapped to expressions of a target logical form language. The expressions of the logical form language are either machine-interpretable representations, which can be potentially grounded in ontologies, or expressive representations of open-domain meaning, such as DRS. This line of research commonly does not explicitly model the compositional derivation process.

On the other hand, prior to the recent advancements in connectionist neural network modelling, the mainstream methodology in computational modelling of semantic parsing was based on using statistical methods to disambiguate interpretations over estimates of the likelihood of derivations (Clark & Curran, 2007; Zettlemoyer &

Collins, 2005, 2007). Such models adopt features that count statistics on symbolic logical form representations and syntactic derivations. Given the transparent syntax-semantics interface in CCG, syntactic disambiguation, in the form of lexical category disambiguation and derivation construction, is also a core component in building an interpretation, since these models have an explicit representation of the compositional process.

Obtaining typed logical forms delivers methodological improvements for both lines of modelling research. In consideration of the sequence-to-sequence models, the thesis shows that a Masked Language Model (MLM) pre-training objective (Devlin et al., 2019) can be devised for DRS which is transparent to the input natural language. Given the DRS variants that are typed, it is proposed that the procedure to mask logical form expressions with MLM is similar to bootstrapping lexicons from lexical templates (Zettlemoyer & Collins, 2005, 2007). This MLM objective is derived from the close correspondence of masking to templatisation over type-assigned DRS expressions, and used to obtain masked examples to pre-train a generalised model from which re-usable word embeddings are distilled that are used in initialisation of the embedding layer of a downstream encoder-decoder semantic parsing model.

In terms of the implications for the classical methods that explicitly model compositionality, the templatisation of the lexical logical form based on the assigned types also permits us to design templatic lexical representations that subsume the output representations in lexical disambiguation. To that end, the thesis presents a methodology to extend supertagging (Bangalore & Joshi, 1999), which is traditionally devised for disambiguating lexical categories for the source natural language, as a lexical item prediction task. The proposed extension is shown to simplify cascaded analysis pipelines, where semantics is projected on a priorly disambiguated parse tree, by re-formalising supertagging as joint disambiguation of lexical syntactic and semantic types.

1.1 Structure of the Thesis

The thesis is structured into two parts. In the first part, from Chapter 2 to Chapter 5, we introduce the linguistic theory and the concepts that are used in formulating the logical form type assignment. The second part, Chapters 6 and 7, presents the main contributions of this thesis to linguistic analysis and computational modelling, accompanied by experimental results.

Chapter 2 presents the linguistic theory that is adopted throughout this thesis, which is CCG. This chapter introduces the syntax-semantics interface in the lexicon, together with combinatory rules that are employed in deriving compositional interpretation.

Chapter 3 defines the types of ambiguities that are encountered while parsing with CCG and reviews the statistical computational models that implement syntactic analysers. The focus of this chapter is on log-linear syntactic parsing models and lexical category disambiguation using supertagging.

Chapter 4 introduces DRT together with its segmented and projective extensions. In this chapter, the compositional derivation of sentence and discourse-level DRS logical

forms is illustrated by encoding lexical semantic types in CCG with DRT.

Chapter 5 is on computational semantic parsing models that map sentences to logical form representations that are then potentially grounded in context. This chapter explains the techniques to bootstrap CCG lexicons over sentences that are annotated for their meaning. Then, statistical semantic parsers for question answering and the models that map open-domain sentences to DRS meaning representations are reviewed.

Chapter 6 is on cross-level type assignment on DRS meaning representations. In this chapter, we first introduce a syntactic description of the DRS logical form language. Then, induce a type system from the grammar that recognises DRS expressions. The DRS type assignment procedure is defined as an extension of parsing DRS expressions. The chapter presents practical applications of logical form typing to semantic parsing by introducing methodologies to templatise and mask DRS logical forms and also to re-formalise supertagging as a lexical item prediction task.

Chapter 7 presents an experiment on using templatised type assigned DRS logical forms as part of an objective to pre-train a generalised model. As part of the experiments, lexical representations are obtained from the pre-trained generalised model to initialise the embeddings in an encoder-decoder DRS semantic parsing model.

CHAPTER 2

COMBINATORY CATEGORIAL GRAMMAR

This chapter presents the linguistic framework that is adopted throughout this thesis. Our linguistic theory of choice is Combinatory Categorial Grammar (CCG). CCG provides the tools to adequately represent the interplay between syntax and semantics in every individual step of the compositional analysis. The foundations are laid by defining the anatomy of a modal variant of CCG and the mechanics of linguistic analysis using its core assumptions, namely the principles of adjacency and categorial government. The overview of CCG provided here follows Steedman (1993) and Steedman and Baldridge (2011).

The aim is to establish a minimum yet sufficient familiarity with concepts that underlie the compositional nature of interpretation, such as derivation through surface structure adjacency and lexical specification of grammatical entities. In the progression towards *Chapter 6 Cross-Level Typing the Logical Form* these concepts are gradually brought into play to illustrate that syntax is the glueing logic to compositionally construct interpretations in various levels of analysis, from lexical, phrasal, sentence-level to discourse.

Section 2.1-Section 2.4 defines the CCG lexicon and syntactic categories and illustrates how syntactic categories combine grammatical constituents using combinatory rules to derive the corresponding compositional interpretation. *Section 2.5* presents the combinatory rule extensions of CCG that provide the expressive power to recognise languages beyond the context-free class. Then, *Section 2.6* shows how CCG is modalised by introducing slash types to restrict its predictive power to attain universally applicable rules.

2.1 The CCG Lexicon

CCG is a lexicalised grammar framework that encodes phonology, morphology, syntax, and semantics at the level of lexicon (Steedman, 1993; Steedman & Baldridge, 2011). The core linguistic specifier in CCG is a collection of grammatical units. A **CCG lexicon** is a finite set of lexical items (or entries) that define the elements of grammar. A **lexical item**, illustrated in (2.1), is a coupled representation of the phonological form, surface form, syntactic category (or syntactic type), and logical form (or semantic type).

(2.1) ‘⟨ phonological form ⟩’ ⟨ surface form ⟩ := ⟨ category ⟩ : ⟨ semantics ⟩

(2.2) ‘ˈ**mərəθən**’ marathon := N : *marathon′*

(2.3) ran := (S\NP)/NP : $\lambda x.\lambda y.$*run′*xy

Coupled specification of these values at lexical-level for every element of grammar enables representation of phenomena such as homophony (e.g. verbal and nominal use of the English ‘*quail*’) and homography (e.g. semantic variation of the English nominal ‘*bow*’) in the lexicon. For the former, distinct lexical items with identical phonological and surface forms but varying syntactic and semantic types are sufficient to capture alternation in use. The latter requires the representation of variations in phonological form and semantics.

This thesis makes the broad assumption that phonological forms (e.g. ‘ˈmərəθən’ represented in IPA in (2.2) and surface forms (e.g. ‘*marathon*’ in (2.2)) are equivalent.[1] It is also assumed that every phonologically realised element of a language has a corresponding lexical entry in the lexicon. This is because, unless otherwise noted, the models and the data that are reviewed and worked with in this thesis are in written modality. Henceforth, orthography will be our distinct specifier of the elements of grammar in the lexicon. The phonological and surface forms are conflated in lexical representation (e.g. in (2.3), instead of specifying the phonological form ‘ræn’ explicitly, only the surface form ‘*ran*’ is provided).

Following Steedman (1993) and subsequent research, the (:=) notation is used to denote syntactic and semantic type assignment. The logical constants are represented with primes to distinguish them from variables. The (:) notation that appears in the type specification of lexical items denotes the correspondence between syntactic category and semantic value. The lexical item in (2.3), for example, reads as "the element of grammar with surface form *ran* has the syntactic type (S\NP)/NP and the associated semantic value $\lambda x.\lambda y.$*run′*xy".[2]

2.2 Syntactic Category

Categories define the syntactic behaviour of lexical items; therefore, they capture how constituents are ordered in the surface structure. They are either **atomic** (or primitive) or **complex**. Atomic categories are the base syntactic types, whereas complex categories are function types. Primitive types are exemplified as S (sentence),

[1] This is a naive assumption, but it helps us in computational modelling of natural languages. The phonological form is the primary modality of external data that a child has access to during language acquisition. However, most efforts in computational modelling of language acquisition and syntactic or semantic parsing are using supervised, semi-supervised or reinforcement learning. These techniques depend on using labelled data in training the models. Data labelling is costly and requires linguistic expertise. So far, datasets that annotate multi-layered representations on phonological and surface forms are minimal or non-existent.

[2] In this thesis, we follow the notation that is adopted by Steedman (1996) to represent syntactic and semantic type correspondence, where an entire syntactic type is associated with an interpretation. This notation differs from the unification-based interpretation notations that associate an interpretation both with the functor and argument syntactic categories while making the mechanics of interpretation unification explicit. As part of the compact notation that we adopt, abstractions of λ-calculus provide the essential mechanics to bind arguments throughout the derivation of the interpretation, as presented in *Section 2.4, p.10*.

N (noun), NP (noun phrase), or PP (prepositional phrase). Hence, the lexical item in (2.2) has an atomic syntactic category, N.

The base case for the complex category has the form X\Y or X/Y. Complex categories define functions from domain Y into a range X. X specifies the result that the functor will yield over an argument of type Y. Both X and Y are atomic or complex categories, meaning that complex categories are recursive functions. Thus, the lexical item in (2.3) has a complex category of the form X/Y, where X is the complex category S\NP and Y is the atomic category NP.

The **slash**, (\) or (/), specifies the expected direction of the argument of the functor. (\) denotes that the functor is seeking an argument to its left. Likewise, (/) determines the argument to be to the right of the functor. For instance, the complex syntactic category NP/N reads as "a functor over an argument with type N to its right yielding a primitive type NP".

Elsewhere, (|) is also used as a shorthand notation to under-specify the direction of the argument (Hockenmaier & Bisk, 2010). X|Y would represent the union of X\Y and X/Y. In this case, the syntactic category X|Y reads as "a functor over an argument with type Y either to its right or left yielding type X".

This notation, also known as the **result leftmost** notation, ensures the leftmost primitive category that appears in a complex category to be the result of the innermost functor (the last evaluated function). The arity of a category is defined as the number of arguments that it seeks to yield a primitive category (e.g. X\Y is of arity 1, whereas (X\Y)/Z is of arity 2). As show in *Section 2.4, p.10*, the result leftmost notation warrants semantic function application to be consistently carried out in left-to-right order.

This definition of the syntactic category inherits the basic notions of syntactic type from AB-calculus (Bar-Hillel, 1953; Kazimierz, 1935) and Lambek calculus (Lambek, 1958). Similar grammar theories that adopt the deduction-style analysis of surface structure over syntactic categories are named the **Categorial Grammars (CG)** in the literature.

2.3 Combinatory Rules

In CG, the derivational process starts with lexical analysis. It assigns a set of lexical items from the lexicon to each lexical constituent over matching surface forms. Syntactic categories of constituents combine via a set of combinatory rules to yield derivations. CG uses the below **function application** rules to combine constituents.

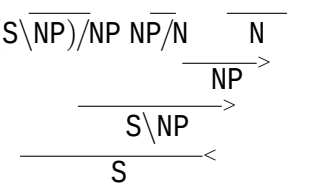
(2.4)
$$\begin{array}{llllll} (>) & X/Y & Y & \Rightarrow & X & \text{(forward function application)} \\ (<) & Y & X\backslash Y & \Rightarrow & X & \text{(backward function application)} \end{array}$$

In its simplest form, CCG is equivalent to CG in that it adopts the same set of application rules in its core. However, it gains further expressive power with an additional set of combinatory rules, which is presented in *Section 2.5, p.12*.

To illustrate, let G be the CCG lexicon that is composed of lexical items in (2.5). Given G, (2.6) is a derivation for the sentence '*Alice ran the marathon*', deduced only using the function application rules.

(2.5) Alice := NP : *alice′*
ran := (S\NP)/NP : $\lambda x.\lambda y.$*run′*xy
the := NP/N : $\lambda x.x$
marathon := N : *marathon′*

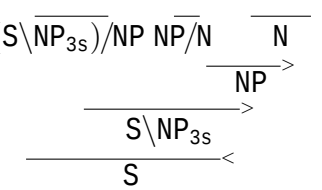
(2.6)



In CCG, the complex category NP/N, which is mentioned in *Section 2.2, p.6*, is the syntactic category that is assigned to English nominal modifiers, such as determiners and quantifiers.[3] English determiners precede the nominal that they modify; hence, they seek a noun argument to their right to yield a noun phrase. The lexical item in (2.5) for the determiner '*the*' combines with the noun '*marathon*' using forward function application as shown in (2.6). The result of this forward function application is a noun phrase ('*the marathon*' with type NP) which is the object of the verb. The transitive verb '*run*' combines with this noun phrase to its right, again using forward function application, which yields a functor that expects a noun phrase to its left (S\NP). Finally, this functor combines with the subject of the sentence '*Alice*' using backward function application, resulting in a sentence (S).

By convention, a set of feature values assigned to the atomic categories represent the number, agreement, case, and other morphological features of constituents. For instance, in (2.7), the morphology of agreement control is marked both on the category of the subject noun phrase (e.g. NP_{3s}) and of the transitive verb (e.g. $(\text{S}\backslash\text{NP}_{3s})/\text{NP}$).

(2.7)



Derivation via combining syntactic categories are related to the re-write rules of context-free grammars (CFG, or context-free phrase-structure grammar; CFPSG). Consider the toy CFPSG in (2.8), which recognises transitive verb phrase constructions of a subset of English sentences with the subject-verb-object (SVO) word order. A CFG, G', is defined as a 4-tuple $G' = (V, \Sigma, R, V_S)$, where V is a finite set of non-terminal symbols (e.g. $\{\text{V}, \text{NP}, \text{S}, ...\}$), Σ is a finite set of terminal symbols (e.g.

[3] Likewise, it could be worked out that, for instance, adjectival adjuncts have the category N\N. An English adjective consumes a noun to its right to yield a noun.

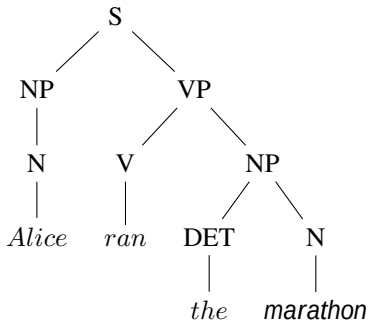
$\{run, the, ...\}$), R is a finite set re-write rules (or productions; e.g. $\{$S $\rightarrow$ NP VP, ...$\}$), and $V_S \in V$ is the non-terminal start symbol (e.g. $V_S =$ S) (Sipser, 1996).

(2.8)

V	$\rightarrow$	ran	
N	$\rightarrow$	$Alice$	| $marathon$
DET	$\rightarrow$	the	
NP	$\rightarrow$	N	| DET N
VP	$\rightarrow$	V	NP
S	$\rightarrow$	NP	VP

The parse tree produced by G' in (2.9) for the above sentence is equivalent to the CCG derivation in (2.6). CCG (and CG) with function application rules can recognise the same set of languages that CFGs can recognise. Hence, function application rules provide context-free power in CCG (Pentus, 1993, 2006).

(2.9)



The syntactic categories of CCG can be regarded as labels of the nodes of the phrase-structure tree. Similarly, there is at least one corresponding lexical item in the CCG lexicon for every terminal symbol of the CFPSG (words or morphemes of the language[4]). Syntactic sub-categorisation is encoded in the lexicon, hence the possibility of a single surface form mapping to multiple lexical items with varying syntactic types. To differentiate transitive and intransitive verbs in CFPSGs, for example, as shown in (2.10), one must capture the syntactic structure of such constructions using distinct non-terminals and production rules (e.g. V_{trans}, $\text{V}_{\text{intrans}}$ and elaboration of the VP production). This is accomplished in CCG by adding a new item to the lexicon with the syntactic category S\NP, as in (2.11).

(2.10)

V_{trans}	$\rightarrow$	ran	
$\text{V}_{\text{intrans}}$	$\rightarrow$	ran	| $bring$
N	$\rightarrow$	$Alice$	| $marathon$
DET	$\rightarrow$	the	
NP	$\rightarrow$	N	| DET N
VP	$\rightarrow$	$\text{V}_{\text{intrans}}$	| V_{trans} NP
S	$\rightarrow$	N	VP

[4] It is possible to define a morphemic variant of CCG, where the elements of grammar are the meaning-bearing morpheme constituents, and corresponding lexical items represent the morpho-syntactic properties of the language. One such example is Bozşahin (2002).

(2.11) ran := $\mathsf{S\backslash NP} : \lambda x.\mathit{run}'x$

Categories do not mark any explicit distinction between lexical and phrasal constituents. Lexical analysis of an intransitive verb (e.g. '*ran*') and a transitive verb that is combined with the object (e.g. the verb phrase '*ran the marathon*' in (2.6)) has the same category, $\mathsf{S\backslash NP}$.

Given its lexicalised nature, CCG is similar to other non-transformational grammar theories such as Tree-Adjoining Grammar (TAG; Joshi and Schabes, 1997), Head-Driven Phrase Structure Grammar (HPSG; Pollard and Sag, 1994), and Lexical Functional Grammar (LFG; Dalrymple, 2001), while fundamentally differing from the rest in how it captures unbounded constructions, such as scrambling and extraction in coordination.

2.4 Deriving Compositional Interpretation

So far, the lexical meaning is represented with λ-calculus expressions. They are composed of variables (e.g. x), constants (e.g. run', which represents a logical constant) or λ-terms (e.g. $\lambda x.\mathit{run}'x$, where $\mathit{run}'x$ is the body of the λ-term, and the variable x that appears in the body is bound since it is λ-abstracted).

Each λ-abstraction is a function definition. In λ-calculus function application is defined as $(M\ N)$, where both the functor M and argument N are expressions (e.g. $M = \lambda x.\mathit{run}'x$ and $N = \mathit{alice}'$). Using β-reduction, the result of function application is obtained by replacing all variables that appear in the body of the λ-term, which are bound by the outermost abstraction, with the argument (e.g. $(\lambda x.(\mathit{run}'x)(\mathit{alice}'))$ yields $\mathit{run}'\mathit{alice}'$ after β-reduction). α-conversion is carried out before β-reduction to avoid collusion of bound variable names (Barendregt, 1984).

There is a close correspondence between function application in semantics and combination through syntactic categories. We extend the definition of combinatory function application rules from (2.4) to account for semantic unification as presented below.

(2.12)
$$\begin{array}{llllll} (>) & \mathsf{X/Y} : f & \mathsf{Y} : g & \Rightarrow & \mathsf{X} : f(g) \\ (<) & \mathsf{Y} : g & \mathsf{X\backslash Y} : f & \Rightarrow & \mathsf{X} : f(g) \end{array}$$

This extension is compliant with CCG's **Principle of Combinatory Transparency**:

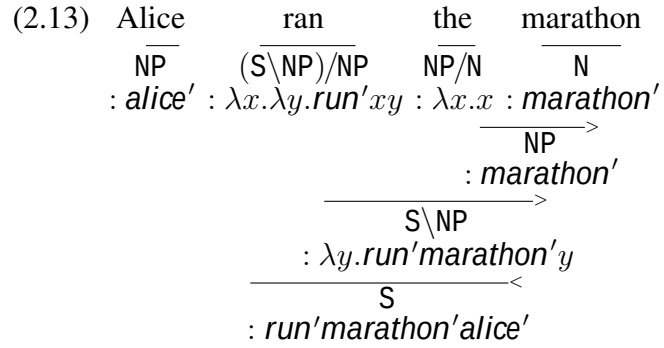
> *The interpretation of a syntactic combinatory rule must be the one that would result from the equivalent combinatorily transparent unification-based interpretation of the rule.* Steedman (1996):13

Using the updated application rule, each derivation now delivers a compositional interpretation, as in (2.13).[5] Combinatory rules define the correspondence of function

[5] Steedman (1996) differentiates *interpretation* from *predicate-argument structure*, where the former is defined as the unreduced logical form that is derived through unification of the interpretations of the constituents, and the

application for both syntax and semantics. Construction of interpretation becomes a part of the derivation. Hence, a transparent syntax-semantics interface is ensured across the board in all levels of analysis.

(2.13)



In this context, the direction of combination of adjacent syntactic categories determine which adjacent semantic value is the functor or argument in interpretation (e.g. in (2.13) the last step of derivation is a backward function application denoting that the semantic value of right-constituent $\lambda y.\mathit{run'marathon'}y$ is the function, whereas the left-constituent semantics $\mathit{alice'}$ is the argument). Therefore, combinatory directionality is only a property of syntax manifesting itself in surface structure. Binding of interpretation via reduction is only carried out in left-to-right order in every step of the derivation over semantic values, which are curried functions.[6,7] This brings us to the first theoretical assumption of CCG, which is the **Principle of Adjacency**:

> *Combinatory rules may only apply to finitely many phonologically realised and string-adjacent entities.* Steedman (1996):5

Note that constants are the atomic units of this logical form representation. Underspecification in semantics is represented with λ-abstractions. If a lexical constituent is assigned a primitive syntactic type, then its associated semantic value is also atomic (e.g. '*marathon*' with atomic syntactic type N has an atomic semantic value of $\mathit{marathon'}$; or the constituent that spans the sentence with type S has a logical form that is only composed of atomic values of $\mathit{run'marathon'alice'}$). If the lexical constituent has a function type, then the arity of λ-abstractions in the logical form matches the arity of the category (e.g. the constituent with category S\NP is of arity 1 and it has the semantic value $\lambda y.\mathit{run'marathon'}y$, which has a single λ-abstraction). This correspondence in syntactic and semantic function abstractions is captured by

latter is the reduced form. In this thesis, we intentionally use these terms interchangeably, although the conceptual distinction is dismissed by doing so. The motivation for this is to align with the terminology that is used in the computational modelling research that is reviewed in *Chapter 5 Semantic Analysis*, where this distinction is mostly omitted.

[6] A curried function is composed of a sequence of functions where each function takes a single argument. Abstractions in λ-calculus are such curried functions. Each λ-abstraction only takes a single argument, hence allowing step-wise evaluation.

[7] The underlying convention in representing the application of function (e.g. $\mathit{run'}$) and arguments (e.g. $\mathit{marathon'}$) in the interpretation is to concatenate them (e.g. $\mathit{run'marathon'}$). Because application is left-associative, the derived interpretation assumes an implicit bracketing (e.g. $(\mathit{run'marathon'})\mathit{alice'}$) that corresponds to an unordered binary tree on which notions of dominance and command apply.

the second theoretical assumption of CCG, which is the **Principle of Categorial Government**:

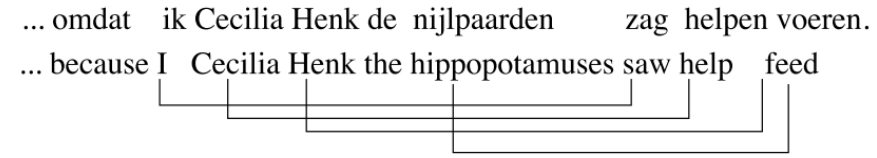
> *Both bounded and unbounded dependencies are entirely determined by lexical syntactic types, which specify semantic valency and cannonical consituent order, and nothing else.* Steedman (1996):5

Although the logical form representation that are presented here has limited expressive power, it illustrates the compositional nature of interpretation. A detailed overview of a representation that adequately captures open-domain meaning is reserved for *Chapter 4 Representing Open-Domain Meaning*.

2.5 Analysis Beyond Context-Free Power

Natural languages belong to the class of **mildly context-sensitive** languages, which is an extension of the original Chomsky hierarchy (Chomsky, 1956). In terms of expressivity, the mildly context-sensitive class lies between the classes of context-free and context-sensitive languages. Languages that belong to this class are characterised by three properties: (limited) cross-serial dependencies; constant growth; and deterministic polynomial time parsing (Weir, 1988).

A well-known example of crossing dependencies is the cross-serial subordinating clause structures of Swiss-German and Dutch (Huybregts, 1976; Shieber, 1985). The example in Figure 2.1 is borrowed from Steedman (2000). Note that the subjects, objects, and verbs of each clause are stacked in succession and therefore in a crossing dependency relation with each other in this example.



'... because I saw Cecilia help Henk feed the hippopotamuses.'

Figure 2.1: An example of Dutch crossing-dependencies in the subordinating clause '*because I saw Cecilia help Henk feed the hippopotamuses*'.

Crossing dependencies of such string sequences cannot be recognised with a single alphabet stacked Pushdown Automaton (PDA), providing sufficient evidence for the weak context-freeness of natural languages. The Embedded Pushdown Automaton (EPDA), which recognises mildly context-sensitive languages, assumes a stack of indexed stack-based memory (Kallmeyer, 2010). The constant growth property of natural languages is ensured due to the proof of their semilinearity (Vijay-Shanker et al., 1987), since all semilinear languages exhibit constant growth. Polynomial time parsing is achieved as an extension of the proof of equivalence between CCG, TAG,

and Head Grammars (HG) to Linear-Indexed Grammars (LIG) by introducing a shared forest representation to chart parsing, which delivers $O(n^7)$ time complexity, n being the length of the parsed sentence. (Vijay-Shanker & Weir, 1990; Vijay-Shanker & Weir, 1993, 1994).

CCG gains its mildly context-sensitive power from how it captures coordination and from additional combinators besides function application, such as composition ($\mathbf{B}$), type-raising ($\mathbf{T}$), and substitution ($\mathbf{S}$). Below, CCG's treatment of conjunction constructions and composition and type-raising rules are introduced.

Coordination is represented by a particular schematic category, $(\mathsf{X}\backslash\mathsf{X})/\mathsf{X}$. Here X is a function schema, the realisation of which depends on the categories of the arguments of the coordinating functor. This category schema allows a conjunct to combine with a constituent to its right, while X is instantiated as the category with which the conjunction combines. From this schema, it is also observed that constituents to the right and left of the coordinating functor can combine with function application iff both arguments have the same category.

An example for coordination is presented in (2.14). The immediate constituent to the right of the conjunct '*and*' is the coordinated verb phrase '*cycled*' with category $\mathsf{S}\backslash\mathsf{NP}$. The coordination category is instantiated as $((\mathsf{S}\backslash\mathsf{NP})\ (\mathsf{S}\backslash\mathsf{NP}))/(\mathsf{S}\backslash\mathsf{NP})$. It combines with the verb phrase to its right using forward function application. Lexical semantics for conjunction, $\lambda p.\lambda k.\lambda z.\mathit{and}'(kz)(pz)$, provide the adequate logical form for coordinating verb phrases at the lexical-level. The semantics of the coordination structure $\lambda z.\mathit{and}'(\mathit{run}'\mathit{marathon}'z)(\mathit{cycle}'z)$ is derived solely using function application, due to the tight coupling of syntax and semantics, similar to the process that is described in *Section 2.4, p.10*.

(2.14)

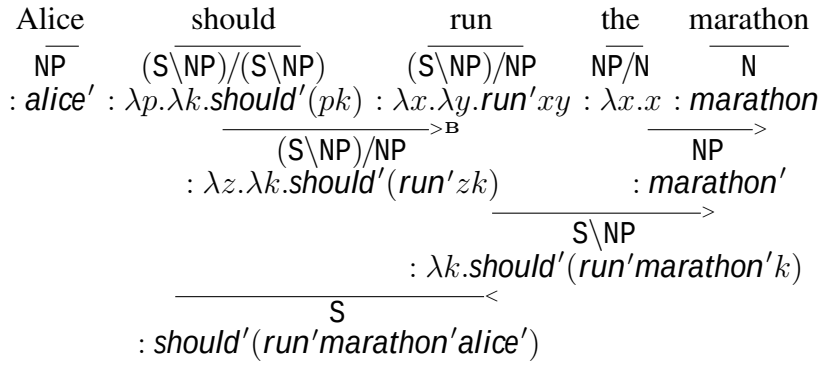
Alice	ran	the	marathon	and	cycled
NP	$(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}$	NP/N	N	$(\mathsf{X}\backslash\mathsf{X})/\mathsf{X}$	$\mathsf{S}\backslash\mathsf{NP}$
$: alice'$	$: \lambda x.\lambda y.run'xy$	$: \lambda x.x$	$: marathon'$	$: \lambda p.\lambda k.\lambda z.and'(kz)(pz)$	$: \lambda x.cycle'x$
			NP (>)	$(\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP})$ (>)	
			$: marathon'$	$: \lambda k.\lambda z.and'(kz)(cycle'z)$	
		$\mathsf{S}\backslash\mathsf{NP}$ (>)			
		$: \lambda y.run'marathon'y$			
		$\mathsf{S}\backslash\mathsf{NP}$ (<)			
		$: \lambda z.and'(run'marathon'z)(cycle'z)$			
	S (<)				
	$: and'(run'marathon'alice')(cycle'alice')$				

The **composition** combinator, adapted from Curry's ($\mathbf{B}$) combinator (Curry et al., 1958), allows contiguous adjacent constituents to compose in non-standard ways. The definition of forward and backward composition is presented in (2.15). To exemplify, for SVO word-ordered English sentences, function application yields (S(VO)) derivations, as in (2.13). Composition also allows ((SV)O) derivations together with type-raising.

(2.15)

$(>\mathbf{B})$	$\mathsf{X}/\mathsf{Y} : f$	$\mathsf{Y}/\mathsf{Z} : g$	$\Rightarrow$	$\mathsf{X}/\mathsf{Z} : \lambda z.f(gz)$
$(<\mathbf{B})$	$\mathsf{Y}\backslash\mathsf{Z} : g$	$\mathsf{X}\backslash\mathsf{Y} : f$	$\Rightarrow$	$\mathsf{X}\backslash\mathsf{Z} : \lambda z.f(gz)$

(2.16)



One use of composition is shown in (2.16), where the modal verb '*should*' combines with the main verb of the clause '*run*' using forward composition, resulting in a category similar to a transitive verb phrase $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}$. The composition rule's associated semantics, $\lambda z.f(gz)$, yields the logical form $\lambda z.\lambda k.\mathit{should}'(\mathit{run}'zk)$, which matches the arity of the category $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}$ while representing the derived interim logical form prior to consumption of subject and object noun phrases.

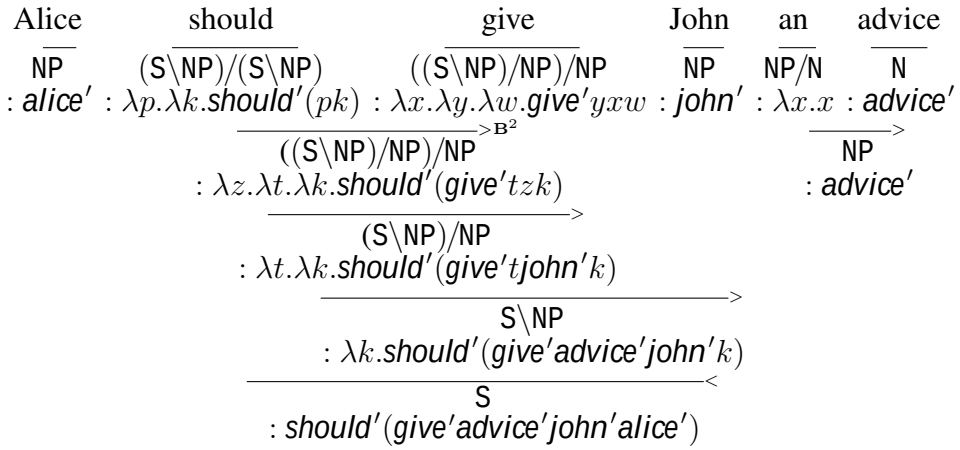
There are two variants of composition, namely **harmonic composition** (B) and **cross composition** ($\mathrm{B}_\times$, or crossing-composition). The composition combinator that is presented above is the harmonic variant, since slash directions in composing categories are the same in forward and backward composition. As defined in (2.17), cross composition does not impose such a restriction on slash directionality. It allows non-adjacent constituents (namely the **permuted arguments**, whose slash directions do not match the functor) to combine.[8]

(2.17)
$$(>\mathrm{B}_\times) \quad \mathsf{X/Y} : f \quad \mathsf{Y}\backslash\mathsf{Z} : g \quad \Rightarrow \quad \mathsf{X}\backslash\mathsf{Y} : \lambda z.f(gz)$$
$$(<\mathrm{B}_\times) \quad \mathsf{Y/Z} : g \quad \mathsf{X}\backslash\mathsf{Y} : f \quad \Rightarrow \quad \mathsf{X/Y} : \lambda z.f(gz)$$

Composition is generalised to compose over an arbitrary number (*n*) of adjacent constituents, as presented in (2.18). This time, the associated semantics of the generalised composition rule λ-abstracts the meaning of composed constituents *n* times in order to match the arity of the yielding category (i.e. $\lambda z_1....\lambda z_n.f(g(z_1...z_n))$).

(2.18)
$$(>\mathrm{B}^n) \quad \mathsf{X/Y} : f \quad \mathsf{Y}|\mathsf{Z_1}|...|\mathsf{Z_n} : g \quad \Rightarrow \quad \mathsf{X}|\mathsf{Z_1}|...|\mathsf{Z_n} : \lambda z_1....\lambda z_n.f(g(z_1...z_n))$$
$$(<\mathrm{B}^n) \quad \mathsf{Y}|\mathsf{Z_1}|...|\mathsf{Z_n} : g \quad \mathsf{X}\backslash\mathsf{Y} : f \quad \Rightarrow \quad \mathsf{X}|\mathsf{Z_1}|...|\mathsf{Z_n} : \lambda z.f(gz) : \lambda z_1....\lambda z_n.f(g(z_1...z_n))$$

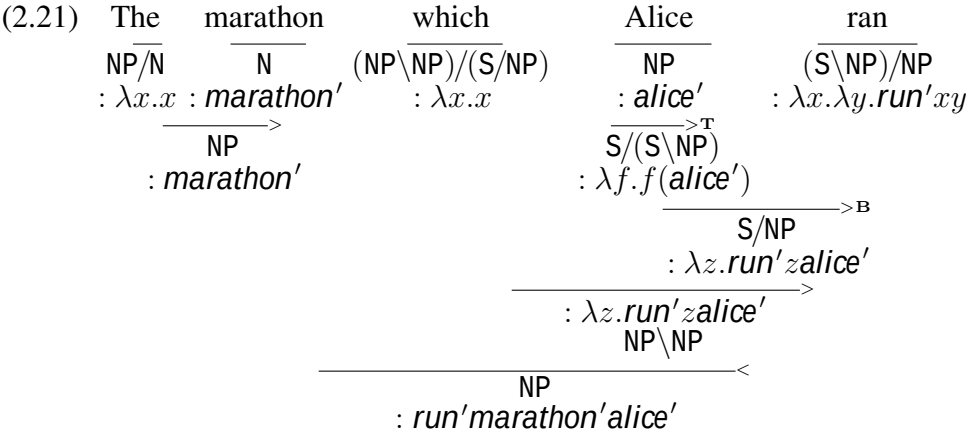
[8] Specific versions of CCG, such as the one that is used in Groningen Meaning Bank and Parallel Meaning Bank datasets, make use of cross composition and its generalised form. *Chapter 4 Representing Open-Domain Meaning* presents an review of these datasets.

(2.19)


For example, in (2.19), second-order generalisation of the composition rule is used to combine the modal verb '*should*' with the ditransitive verb '*give*', while yielding the category $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP})/\mathsf{NP}$, which preserves the number of arguments that the lexical verb '*give*' seeks.

Type-raising is a unary rule that converts arguments into functors that expect functors over such arguments. Forward and backward type-raising rules are defined in (2.20). Similar to the conjunction category, T refers to a schematic category in these definitions. It comes from the set of functions that have X as the domain (e.g. if X is NP, then $\mathsf{T} \in \{\ \mathsf{S}, \mathsf{S}\backslash\mathsf{NP}, (\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}\ \ldots\ \}$). Category X is only allowed to be a primitive type. It is helpful to assume that this rule is only applied to lexical categories to limit its derivational productivity.

(2.20)
$$(>\mathbf{T}) \quad \mathsf{X} : g \quad \Rightarrow \quad \mathsf{T}/(\mathsf{T}\backslash\mathsf{X}) : \lambda f.fg$$
$$(<\mathbf{T}) \quad \mathsf{X} : g \quad \Rightarrow \quad \mathsf{T}\backslash(\mathsf{T}/\mathsf{X}) : \lambda f.fg$$

(2.21)


The frequent use of the type-raising rule in English is to turn subject noun phrases into functors that seek a verb phrase to their right.[9] It is exemplified in (2.21), where the noun phrase '*Alice*' in the subject position is turned into a function using forward

[9] Type-raising is also used as a mechanism to capture the morpho-syntactic variation of agreement morphology. See Kiss and Alexiadou (2015) for an example of control agreement morphology in Modern Standard Arabic.

type-raising while realising T as S and yielding the category $\mathsf{S}/(\mathsf{S}\backslash\mathsf{NP})$. Note that prior to type-raising, the lexical meaning of '*Alice*'was a logical constant, alice', which also turned into a function abstraction, $\lambda f.f(\mathit{alice}')$ due to the semantics of forward type-raising.

2.6 Modal CCG

Making unrestricted use of all combinators that are presented in *Section 2.5, p.12*, and function application gives CCG an abundance of predictive power that yields derivations with ungrammatical word order. The solution to restricting the generative power of CCG to achieve a universal theory of grammar that does not assume language-specific bans over the use of certain combinators is to modalise the slashes (Baldridge, 2002; Baldridge & Kruijff, 2003; Jacobson, 1992). In other words, we assign types to slashes in the form of features that introduce lexical control in combinatory rules.

The feature set, as it is defined in Baldridge (2002) and Baldridge and Kruijff (2003), is composed of four slash types, which have the following behaviour:

(2.22)
- $\cdot$ associative and permutative
- $\diamond$ associative and non-permutative
- $\times$ non-associative and permutative
- $*$ non-associative and non-permutative

Figure 2.2 shows the type-hierarchy of slash modalities. Types inherit their properties from top to bottom in the hierarchy. That means the most restrictive type is $(*)$ and the most permissive type is $(\cdot)$. Types $(\diamond)$ and $(\times)$ partially inherit restrictive properties from their supertype $(*)$. Likewise, $(*)$ inherits all the permissive properties of $(\diamond)$ and $(\times)$. Here, associativity refers to the degree of allowed re-bracketing in a derivation provided by composition and type-rasing in coordinated structures. Permutativity refers to the re-ordering of permuted arguments when combined with cross composition.

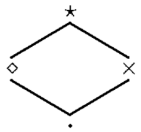
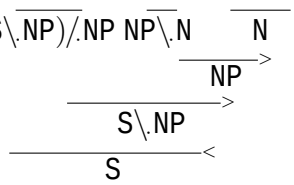


Figure 2.2: Hierarchy of slash types in modal CCG.

We now assign a mode to each slash that appears in syntactic categories, such as the permissive slash modes $(\cdot)$ in lexical categories of (2.24). Since $(*)$ is the supertype of all other types, function application rules are now re-defined as in (2.23). This type assignment allows all categories with slashes of any type to combine with function application.

(2.23)
$$(>) \quad \mathsf{X}/_{\star}\mathsf{Y} : f \quad \mathsf{Y} : g \quad \Rightarrow \quad \mathsf{X} : f(g)$$
$$(<) \quad \mathsf{Y} : g \quad \mathsf{X}\backslash_{\star}\mathsf{Y} : f \quad \Rightarrow \quad \mathsf{X} : f(g)$$

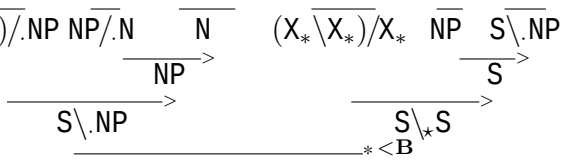
(2.24) Alice ran the marathon



Similarly, we define the coordination category as $(X_*\backslash X_*)/X_*$ and update our definition of harmonic composition as in (2.25). With this update, harmonic composition only allows categories that have slashes with mode ($\diamond$) or its subtype ($\cdot$) to compose, which disallows illicit coordination derivations, such as the one shown in (2.26).

(2.25)
$$(>\mathbf{B}^n)\quad X/_\diamond Y : f \quad Y|_\diamond Z_1|_\diamond ...|_\diamond Z_n : g \quad \Rightarrow \quad X|_\diamond Z_1|_\diamond ...|_\diamond Z_n : \lambda z_1....\lambda z_n.f(g(z_1...z_n))$$
$$(<\mathbf{B}^n)\quad Y|_\diamond Z_1|_\diamond ...|_\diamond Z_n : g \quad X\backslash_\diamond Y : f \quad \Rightarrow \quad X|_\diamond Z_1|_\diamond ...|_\diamond Z_n : \lambda z_1....\lambda z_n.f(g(z_1...z_n))$$

(2.26) Alice ran the marathon and John cycled



Cross composition is also restricted to only compose over categories that have slash types ($\times$) and ($\cdot$), as in (2.27). The cross composition's non-order preserving nature, which allows permutations over arguments, is limited. For instance, the heavy NP-shift of English (i.e. in cases where an adverb appears in its non-canonical position, between the main verb of a clause and the object of the verb) is captured without imposing an English-specific ban on $<\mathbf{B}_\times$(Baldridge & Kruijff, 2003).

(2.27)
$$(>\mathbf{B}_\times)\quad X/_\times Y : f \quad Y\backslash_\times Z : g \quad \Rightarrow \quad X\backslash_\times Y : \lambda z.f(gz)$$
$$(<\mathbf{B}_\times)\quad Y/_\times Z : g \quad X\backslash_\times Y : f \quad \Rightarrow \quad X/_\times Y : \lambda z.f(gz)$$

Finally, the type-raising rule is also re-defined to account for higher-level modal control, as in (2.28). Note that type-raising only applies to atomic categories, which do not contain slashes. Nevertheless, the syntactic category in the codomain of the type-raising rule includes slashes. The variable i that is on the output category slashes in forward and backward type-raising denotes that the mode of both slashes are of any type from the type hierarchy, but they need to match. The modal variable i is realised as a type according to the slash mode which appears in the argument category $(T\backslash_i X)$ that the raised function combines.

(2.28)
$$(>\mathbf{T})\quad X : g \quad \Rightarrow \quad T/_i(T\backslash_i X) : \lambda f.fg$$
$$(<\mathbf{T})\quad X : g \quad \Rightarrow \quad T\backslash_i(T/_i X) : \lambda f.fg$$

Given the modal control through slash types, the combinatory rules of CCG become genuinely universal. Lexical control restricts the generative power of CCG by solely defining an adequate set of lexical categories per language that only allows derivations with grammatical word order that the language permits.

CHAPTER 3

SYNTACTIC ANALYSIS

This chapter presents a selection of prominent modelling literature that utilises CCG in syntactic parsing. The focus is mainly on two lines of research. First, the parameterised log-linear models that implement CCG dependency parsing. Second, the flavor of models that adopt a lexical category tagging technique, supertagging, which was initially devised for parsing with TAGs. Supertagging models are commonly employed as precursors to syntactic parsers, they minimise the lexical ambiguity and narrow down the search space of the cascaded parser.

By syntactic parsing, it is meant that these models do not take into account interpretation as part of the derivational process. The task is defined as finding the set of all possible (or most probable) syntactic derivations for a given sentence. *Chapter 5 Semantic Analysis* reviews the models that provide an interpretation together with syntactic analysis. Hence, this chapter serves as a prerequisite for some of the concepts that are mentioned there, such as bootstrapping CCG lexicons as part of parameter estimation.

The chapter starts by defining the types of ambiguity that arise in syntactic parsing with CCG. Then, *Section 3.2* briefly reviews the human-annotated and automatically generated resources that are used to provide supervision during training of the parsers. *Section 3.3* presents a log-linear parsing method and mention the techniques that are used in further constraining the search space and controlling the combinatorial explosion, namely normal-form parsing and beam search. *Section 3.4* and *Section 3.5* are about symbolic and neural variants of non-constructive and constructive CCG suppertagging models.

3.1 Ambiguity in CCG

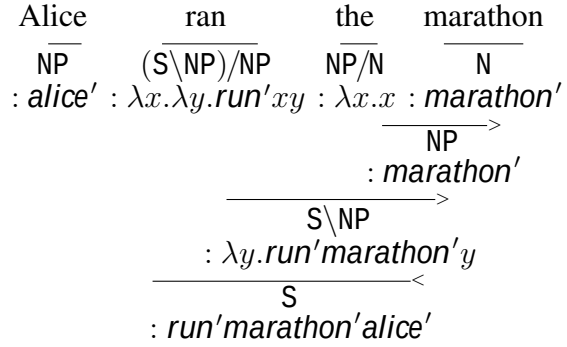
Natural language is ambiguous. While parsing with CCG, two major types of ambiguity are encountered: lexical and spurious ambiguity.

Lexical ambiguity arises when a surface form is associated with multiple lexical items in the lexicon. The syntactic analysis process with CCG starts with finding a match between a set of lexical items from the lexicon and the lexical constituents of a given sentence. Since CCG is radically lexicalised under the Principle of Categorial Government, syntactic categorisation and variation in interpretation are encoded in the

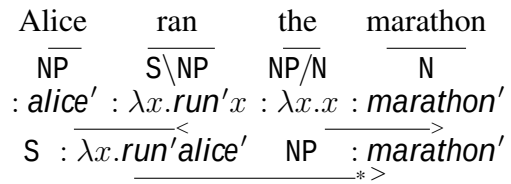
lexicon (as illustrated in *Section 2.4, p.10*). Given our prior assumption from *Section 2.1, p.5* on using surface form as the sole distinctive specifier of the grammar elements, all lexical items with a matching orthographic surface form for a lexical constituent are associated with it throughout lexical analysis.

Consider the transitive and intransitive categorisation of the verb '*run*' in (3.1). Both analyses for the lexical constituent are applicable if they are part of the lexicon and therefore permit the possibility of yielding distinct derivations stemming from both uses. In this example, (3.1)-b does not lead to a derivation that spans the sentence, since the intransitive verb category only seeks a subject noun phrase to its right, and hence the object noun phrase is left standalone. Though, it is observed from the example that given a wide-coverage lexicon for the target language, even while keeping the set of combinatory rules minimal, lexical ambiguity can inflate the set of derivations obtained for certain sentences.

(3.1) a.

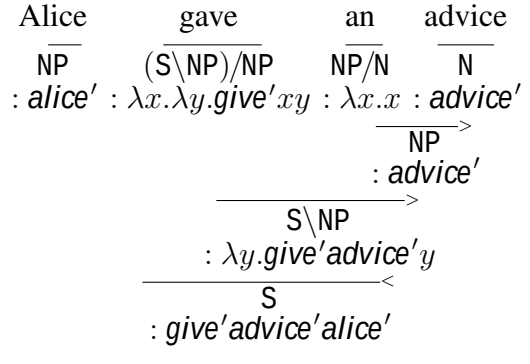


b.

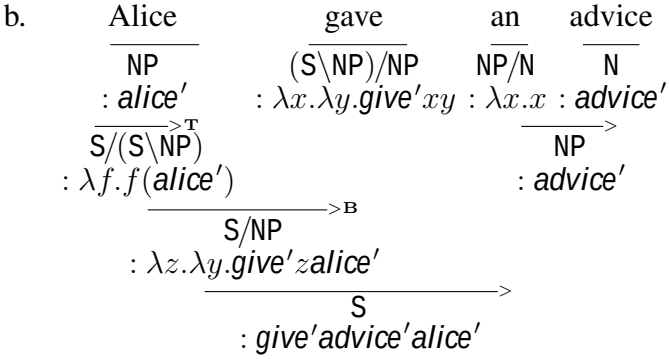


Spurious ambiguity occurs when multiple dissimilar derivations yield the same interpretation. In CCG, one frequent source of spurious ambiguity is the application of type-raising and composition rules in succession (e.g. mostly on lexical noun phrase constituents). Here, type-raised construction leads to a derivation where the transitive verb is first combined with its subject through harmonic composition, and it takes the object as its second argument. Both analyses yield a derivation that leads to the sentence category (S) while differing in their internal structure.

(3.2) a.



b.



Syntactic parsers, which are reviewed through this chapter, deal with both types of ambiguity. The task is defined as given a sentence that is composed of a series of functionally meaningful tokens (lexical constituents, e.g. $\{Alice, gave, an, advice\}$) and a CCG lexicon with enough coverage to analyse the tokens of the given sentence, we would like to find the most probable derivation by combining categories of the adjacent constituents using an assumed set of combinators that leads to the sentence category S while disambiguating lexical and spurious ambiguities.

3.2 CCG Treebanks

A treebank is a resource that consists of corpora of representative natural language sentences, which are usually sampled from long form text, that are annotated with certain morpho-syntactic features. Conventionally annotated features include syntactic categories or parts-of-speech (POS), lexical-level morphological features, and hierarchical constituency structures or syntactic dependency relations. Treebanks are used for providing supervision in training syntactic parsers under supervised learning schemes, such as the models that are reviewed in the rest of this chapter. The domain (or genre) of the annotated text varies depending on the coverage of the resource and its intended use. It is common to observe that the corpora that are used in building such treebanks include texts from news articles, literature, and domain-specific technical manuscripts.

Penn Treebank (PTB; Marcus et al., 1993) is a well-cited example in the literature that annotates English sentences for their underlying constituency tree structures. Its corpus is composed of newspaper text that is sampled from the Wall Street Journal (WSJ). Likewise, the Universal Dependencies project (Nivre et al., 2016; Nivre et al., 2020) consists of monolingual treebanks in various languages that are annotated for syntactic dependencies using a label set that is shared across annotations of all covered languages.

CCGBank (Hockenmaier, 2003; Hockenmaier & Steedman, 2007) is a large-scale, wide-coverage treebank that annotates lexical constituents of individual sentences with CCG syntactic categories and also provides sentence-level derivations. It is semi-automatically converted from PTB and contains annotations for 48,934 English sentences. This corresponds to 99.44% of the PTB corpora, while the remaining PTB sentences are discarded from CCGBank since valid CCG annotations cannot be recovered for them by means of automatic conversion. The conversion algorithm

takes a PTB sentence, lexical POS, and phrase-structure tree annotations as input and outputs a head-dependency structure annotation in the form of 4-tuple, an example of which is as follows:

(3.3) $\langle ran, (\mathsf{S[dcl]\backslash NP_1)/NP_2}, 1, marathon\rangle$
$\langle ran, (\mathsf{S[dcl]\backslash NP_1)/NP_2}, 2, alice\rangle$

Dependency annotations such as above encode the head word of the lexical category (e.g. ran), functor category with dependent information encoded as subscript (e.g. $(\mathsf{S[dcl]\backslash NP_1)/NP_2}$), argument slot (e.g. 1 or 2), and the head word of the argument (e.g $marathon$) as items of the tuple.

All sentences of PTB that do not contain any null elements in their phrase-structure tree are mapped to such dependency annotations with a conversion algorithm. The null elements are incompatible with CCG analyses, hence they are handled with special procedures. The baseline conversion algorithm first determines the constituent types (e.g. differentiates the heads, complements, and adjuncts of the sentence), binarises the phrase-structure trees, assigns CCG categories to the nodes of the binary tree, and finally specifies the dependency structure over the tree that encodes node-specific syntactic categories.

In order to evaluate the out-of-domain performance of parsing models, there are two additional treebanks that annotate text with CCG from domains other than news. These are the Bioinfer and Wikipedia datasets (Honnibal et al., 2009; Rimell & Clark, 2008).

The Bioinfer dataset is built on top of the corpus that is crafted by Pyysalo et al. (2007) which is composed of biomedical web content and research papers. The annotated text is highly representative of domain-specific terminology and therefore poses a challenge for parsers that are only trained on newspaper text in terms of limited lexical overlap across domains. Rimell and Clark (2008) sampled 1,000 sentences from the Bioinfer corpus and manually annotated the lexical constituents of those sentences with CCG categories. The resulting dataset does not annotate gold standard CCG derivations for these sentences, but only the syntactic categories, since the resource was originally developed to test domain adaption of CCG parsers through domain-specific re-training of their integrated supertaggers. The resource is commonly used by the subsequent supertagging literature to test tagging accuracy across domains.

An unbiased sample from a dump of Wikipedia entries yields sentences that are lengthier than samples obtained from domains like news and literature (Kayadelen et al., 2020). Moreover, Wikipedia sentences cover a variety of topics with a rich representation of named entities and are therefore regarded as an adequate test bed to evaluate parsing accuracy in open-domain text. To this end, Honnibal et al. (2009) manually annotated a randomly sampled set of 200 Wikipedia sentences with gold standard CCG categories and also derivations to investigate the effects of domain adaptation of their log-linear parsing model. This dataset is mostly used as a cross-domain evaluation set for syntactic parsing and supertagging in the following literature.

3.3 Log-Linear Syntactic Parsing

Due to the binary nature of derivation using assumed combinators of CCG (except the type-raising rule that is usually only applied to the lexical constituents), bottom-up parsing algorithms are considered suitable to implement psychologically plausible parsers for CCG (Steedman, 2000). Early efforts adopt dynamic programming methods such as the CYK parsing algorithm (Clark, 2002; Clark & Curran, 2003, 2004b), whereas latter work demonstrate that shift-reduce parsing performs equally well for CCG (Zhang & Clark, 2011). Together with the advancements in deep learning, neural models have also started to emerge in research that presents sequence-to-sequence syntactic parsers based on LSTM architectures with varying disambiguation methodologies (Lewis & Steedman, 2014a; Xu, 2016).

Among these models, one line of research, commonly referred to in the literature as the C&C parser, is reviewed in this section (Clark & Curran, 2004b, 2007). The parser implements a log-linear model using the supervision from head-dependency annotations, such as those that are shown in *Section 3.2, p.21*. The model defines probabilities for a given dependency structure as the following conditional probability:

$$P(\pi|S) = \sum_{d \in \Delta(\pi)} P(d, \pi|S) \tag{3.4}$$

Here, the dependency structure, π, given a sentence, S, is approximated as the sum of all derivations, d, from the set of derivations, $\Delta(\pi)$, that lead to π. The dependency structure, π, is a set of 4-tuple head-dependency annotations. In this context, the maximum entropy parsing model is defined as follows:

$$P(w|S) = \frac{1}{Z_S} e^{\sum_i \lambda_i f_i(w)} \tag{3.5}$$

Above, w is a syntactic parse from the set of all possible parses for the sentence, S, where f_i is a single feature among the set of features that are defined by the model, λ_i is the corresponding weight for this feature, and Z is a normalisation constant. The syntactic parser, w, is defined as a 2-tuple of derivations, d, and dependency structures, π. The features and weights parameterise the model. Each feature counts the occurrence of either a certain structure in derivations, or a certain dependency relation in the dependency structure. The feature weights are estimated using discriminative estimation methods over the likelihood of the dependency structures of the examples in the training set.

When disambiguating the syntactic parse of wide-coverage natural language input with such a log-linear model, an increase in the search space and parse complexity is observed due to high degrees of lexical and spurious ambiguity. Some common techniques that are employed by the literature to constrain the search space and to increase the parser efficiency include introduction of CYK chart packing and beam search in parser implementation, as well as controlling spurious ambiguity with normal-form rules.

In CYK parsing, all possible derivations for a given span, which are obtained through application of the assumed set of combinatory rules to the constituent syntactic categories, are added to the chart cell, which corresponds to the parsed span, as standalone chart entries. **Packed charts** (Miyao & Tsujii, 2002) reduce the entries in populated chart cells to minimise the number of combinatory rule applications on chart entries by conflating the derivations that yield the same syntactic category for a given span. If multiple derivations yield the same category for a span, then individual derivations that lead to that category is traced with back pointers. In addition, while using **beam search** (Bodenstab et al., 2011) throughout inference, only those chart entries that correspond to the derivations whose probability is above a certain beam threshold is kept in the chart, while pruning derivations with lower likelihood. Beam search can be applied for any level of the derivation with varying strictness, hence reducing the overall chart size to increase parsing efficiency.

In **normal-form parsing** (Eisner, 1996; Hockenmaier & Bisk, 2010) the spurious ambiguity that results in multiple derivations with identical interpretations is eliminated to attain a single canonical derivation that leads to the interpretation. That is achieved by restraining combinatory rule applications that lead to spurious ambiguity. For instance, Eisner (1996) introduces a normal-form constraint on forward composition ($>\mathbf{B}$) assuming a CCG that does not accommodate type-raising. The combinatory rule restriction suggests that the forward application ($>$) and composition cannot be applied to any category that is derived through forward composition ($>\mathbf{B}$). Likewise, in Eisner's normal-form, it is also not permitted to apply successive forward compositions in a derivation. Hockenmaier and Bisk (2010) extends Eisner's normal-form rules to a CCG that includes type-raising and generalised composition as part of the assumed combinatory rule set.

The log-linear model and the search space restriction techniques that are reviewed in this section make up the basis for the semantic parsing models that are presented in *Section 5.2, p.58*. There, the lexical semantic types are also specified as part of lexical items in the lexicon, and log-linear parsers are modelled over the likelihood of the compositionally constructed interpretation.

3.4 Non-Constructive Supertagging

Supertagging is a method that is devised for parsing with Lexicalised Tree-Adjoining Grammars (LTAG; Bangalore and Joshi, 1999). It exploits the concept of lexicalisation of the grammar, and hence it is also used in parsing with other lexicalised grammar formalisms, such as CCG. The underlying idea of supertagging is to eliminate lexical ambiguity to the best possible extent by first formalising a tagging task in the lexical scan phase of parsing prior to parse tree construction, so that a dynamic programming algorithm like CYK is used to deterministically build the parse tree over disambiguated syntactic categories in an efficient manner.

This tagging task is defined as predicting a supertag from a tag set T for each token, w, of a sentence, $S = \{w_1, \ldots, w_n\}$, where n is the number of tokens in the sentence. Therefore, supertagging is considered an approach that is almost parsing while delivering improvements in accuracy and time and space complexity of syntactic parsers for

lexicalised grammar formalisms.

A **supertag** is defined as a rich specification of complex constraints on a local context (Bangalore & Joshi, 1999). They should not be confused with POS tags since there could be a one-to-many mapping between the set of POS tags and supertags.[1] Within the context of LTAG parsing, the set of supertags, T, is obtained by aggregating the primitive elements of the grammar, recursive and non-recursive variants of elementary trees, that are assigned to lexical items in the lexical scan phase of parsing.

Elementary trees are complex descriptions of syntactic and semantic constraints imposed on a constituent, namely an anchor in LTAG terms. They can be deterministically converted to the lexical items of CCG. LTAG defines two operators, substitution and adjunction, to combine elementary trees of adjacent constituents (Joshi & Schabes, 1997). An LTAG derivation is then defined as the process of combining lexical elementary trees using these two combination operators to obtain a syntactic parse of a sentence. In this context, the supertagging problem is the first step of syntactic analysis that aims to resolve the ambiguity in assigning elementary trees to lexical constituents in cases where multiple elementary trees are associated with a constituent due to ambiguity when the context of the constituent is taken into account.

Bangalore and Joshi (1999) present the first empirical results on supertagging over the sentences from PTB. The tag set is composed of 300 elementary trees, which are derived from the XTAG grammar (XTAG Research Group, 1995). XTAG grammar is manually crafted, therefore yielding a tag set that is small in size with higher quality specifications, which are expected to yield improved accuracy over semi-automatically generated tag sets. They present an n-gram Hidden Markov Model (HMM) supertagging model. The baseline is to train a unigram model that estimates the lexical preference of each lexical constituent, not conditioned on context, and solely predicts a supertag for the given lexical item based on the highest observed frequency tag for that item in the training data. The model is potentially extended to arbitrarily large n-gram windows in training and inference, where Bangalore and Joshi (1999) use the following trigram model for their experiments:

$$\hat{T} = argmax_T \prod_{i=1}^{n} P(t_i|t_{i-2}, t_{i-1}) * \prod_{i=1}^{n} P(w_i|t_i) \quad (3.6)$$

Here, t_i is the supertag from the tag set T for the word w_i, where i is the position of the word with respect to the sequence of tokens that compose the sentence S. Note that the first term is a trigram approximation of the probability of predicting a supertag given a context window of two preceding tags that are assigned to the lexical constituents that are on the left. Similarly, the second term approximates the probability of a word given the supertag that is assigned to it. In inference time, in order to approximate a prediction for unseen tokens, smoothing techniques are applied after training to redistribute frequencies from observed hypotheses to unobserved hypotheses. Bangalore and Joshi (1999) use the Katz's back off model (Katz, 1987).

Clark (2002) is the first research on incorporating supertagging into CCG parsing. The tagger is an adaptation of a feature-based POS tagger (Ratnaparkhi, 1998), which

[1] As illustrated in *Section 2.3, p.7* the syntactic categories that describe English transitive and intransitive verbs in CCG are not the same (S\NP vs. (S\NP)/NP), whereas both such verbs are labelled with the same POS tag VERB.

is a maximum entropy model that estimates the probability of assigning a syntactic category from the lexical category tag set to a word given the context of the word. Distinctive from the LTAG supertagger of Bangalore and Joshi (1999), supertag set is obtained from normal-form derivations of the CCGBank. All syntactic categories that are assigned to tokens in the CCGBank are aggregated to make up the tag set. In what follows, supertags are commonly referred as syntactic categories, since in CCG supertagging prediction space is the set of unique syntactic categories that are encountered in a CCG lexicon.

In comparison to Bangalore and Joshi (1999), the category tag set of Clark (2002) is 4 times larger in size (1,026 unique lexical syntactic categories are observed in CCGBank annotations). A larger category set stems from the fact that CCGBank is constructed through semi-automatic conversion of PTB POS and phrase structure annotations, which results in syntactic categories that occur infrequently in CCGBank annotations. A larger category set translates to a larger search space for the tagging model and is hypothesised to harm the tagging accuracy. Therefore, a common practise in the literature is to apply pruning to the supertag set to minimise its size, usually using a frequency cut-off, where the category frequencies are computed with respect to a representative corpora (Chen & Vijay-Shanker, 2000).

The conditional model that Clark (2002) uses for supertagging is as follows:

$$P(t|h) = \frac{1}{Z(h)} e^{\sum_i \lambda_i f_i(t,h)} \tag{3.7}$$

In the above log-linear form, the probability of the category, t, that we are predicting given a context window, h, is approximated over arbitrary features, f, and the weight of the feature, λ, where Z is a normalisation constant given the context. The size of the context window is an experimental variable constrained by the feature definitions. Features are contextual predicates. They could simply be conditioned on ad-hoc lexical predicates such as whether the orthography of the lexical constituent that we are tagging is a sought form (e.g. the word is a determiner or a digit), or they are complex in terms of being conditioned on the features of the context tokens (e.g. the word we are tagging is a successor of a determiner).

Clark and Curran (2004a) present a multi-tagger that is based on the above log-linear model. A multi-tagger assigns all possible categories that are predicted by the model with a probability above a certain beam threshold, β. Multi-tagging potentially results in lexical ambiguity in comparison to predicting a single category for each token as part of supertagging. However, the empirical results of Clark and Curran (2004a) demonstrate that overall parsing accuracy and β are inversely related. Therefore, they present the trade-off between accuracy and time and space complexity in parsing under varying beam strictness regimes of the supertagger. One common strategy is to start with a narrower beam size, while the base case being the single-tagger, and if no derivation is found then gradually relax the search space by increasing the beam with a lower β threshold.

Follow up research formalises the problem as a sequence-to-sequence tagging task and uses connectionist neural network models and deep learning techniques to optimise on

the training data before evaluating model performance on unseen examples. Under the sequence-to-sequence paradigm the task is formalised as predicting a sequence of tags $T = \{t_1, \ldots, t_n\}$ for an input sequence of words that compose a sentence $S = \{w_1, \ldots, w_n\}$, where n is the input and output sequence length defined by the number of tokens in the sentence.

Lewis and Steedman (2014b) present the early findings of using pre-trained word embeddings to encode tokens of the input sequence, which has a positive impact on parsing accuracy while introducing semi-supervision to the disambiguation task and reducing the dependency on supervision through labelled training data. Instead of the hand-crafted linguistically motivated contextual features of Clark and Curran (2004a), they use word embeddings together with some shallow contextual structural features (e.g. n-character suffix or capitalisation pattern of the preceding k words that form up the context). They experiment using two models. The first is a 2-layer feed-forward neural network model that is inherited from POS tagging (Collobert et al., 2011); the latter is a conditional random field model (CRF; Turian et al., 2010). Word embeddings are static lookup tables with the dimension $V \times D$, where V is the size of the vocabulary that the models use (or size of the assumed CCG lexicon that can parse the input sentences) and D is the dimension of each word embedding vector. Both models are used as classifiers, meaning that given a window of context with m tokens, where m is usually smaller than the number of tokens in a given sentence, the supertagging model classifies each token with respect to its syntactic category.

Xu et al. (2015) address one prominent limitation posed by the models presented by Lewis and Steedman (2014b), that is the limited size of the context window of both the neural network and CRF models, which does not span the full context of the sentence during training and inference. Xu et al. (2015) is the first work that models supertagging with a Recurrent Neural Network (RNN), specifically an Elman network (Elman, 1990). The Elman network is composed of three layers. The first is the input layer, Xu et al. (2015) use pre-trained word embedding (Turian et al., 2010) to encode the natural language input as part of the input layer. The second layer is the fully connected hidden state layer that implements recurrent connections to previous hidden states that aggregate a representation of the left context up until the current prediction state. The final layer is the output layer (usually implemented as a softmax layer), which is a representation of the probability of the predicted category for a given state, dependent on the current input state and context representation. RNNs' capability of representing full precursor context enables them to outperform shallow classifiers that Lewis and Steedman (2014b) present on the supertagging task.

Lewis et al. (2016) and Vaswani et al. (2016) model the task using LTSMs (Hochreiter & Schmidhuber, 1997), specifically bi-LSTM variants that can read the input in both left and right directions (Schuster & Paliwal, 1997). Bi-LSTMs are specifically capable of dealing with long range dependencies that the underlying CCG permits as they represent the bi-directional context of a word that is tagged in a given time step. Vaswani et al. (2016) simplify the model by only using word embedding features while encoding the input, discarding the shallow contextual n-character suffix and capitalisation features, and employing greedy decoding over the softmax output layer. Even with such simplifications, bi-LSTMs outperform feed-forward and Elman networks in CCG supertagging. In addition, Lewis et al. (2016) implement the A* CCG parsing algorithm (Lewis & Steedman, 2014a) on top of the bi-LSTM softmax

layer, which delivers a complete parse tree over the sequence of tags that are output by the underlying LSTM supertagger.

Tian et al. (2020) build on the evidence that contextual information improves supertagging and parsing accuracy through the use of LSTMs and leverage the approach by using a Graph Convolutional Network (GCN; Scarselli et al., 2008) to learn the span-adjacency information while tagging. Conventionally, GCNs are used in syntactic parsing on top of a dependency parse by benefiting from the dependency edge information that the dependency parse delivers to weight the structurally related chunks. However, in the case of CCG supertagging, such a dependency parse is not always available. Tian et al. (2020) use an unsupervised approach as a proxy to infer adjacency tables of the GCN. Prior to tagging, they obtain a lexicon of all possible n-grams that appear in a reference corpus and only mark the span of n-grams that appear both in the sentence that is being tagged and the pre-populated lexicon as the relevant chunks of the input. Since this approach delivers only the frequently observed chunks but not their dependency relationships, attention is incorporated into GCN to span over the marked chunks and learn their relatedness weights during the learning stage while using attention as a proxy. This approach further illustrates that elaborate representation of the context has a direct positive impact on supertagging and parsing accuracy.

3.5 Constructive Supertagging

All the models that are presented in *Section 3.4, p.24* are trained to predict a CCG syntactic category for each word in the input sentence, without taking into account the internal structure of the category during training time. As shown in *Section 2.2, p.6*, CCG syntactic categories can either be primitive (e.g. NP) or complex (e.g. $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}$). In the **non-constructive** form of CCG supertagging, all possible syntactic categories, no matter whether they are primitive or complex, are holistically part of the supertag set which makes up the output space. Under sequence-to-sequence modelling paradigm, the **constructive** variant of the task is formalised as predicting the individual tokens of deconstructed and linearised supertags (e.g. the model would predict a linearised series of tokens $\{(, \mathsf{S}, \backslash, \mathsf{NP},), /, \mathsf{NP}\}$ that represent the category $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}$). This approach entails that the model also learn the internal structure of the category and the auxiliary representational conventions such as proper bracketing to generate function categories that can be meaningfully curried.

Bhargava and Penn (2020) is the first research that present such a constructive supertagging model. Similar to Lewis et al. (2016) and Vaswani et al. (2016), they use a bi-LSTM model. However, they present an encoder-decoder model, where both the encoder and the decoder stacks are bi-LSTMs, which read the input and output sequences bi-directionally. This is different from Lewis et al. (2016) and Vaswani et al. (2016)'s encoder-only approach, where a softmax output layer is projected over a bi-LSTM encoder stack. The output sequence is composed of flat linearised category representations, which are obtained by tokenising the syntactic category strings into their functional units (e.g. each parenthesis, slash or primitive category acts as a single token). An encoder-decoder bi-LSTM model is shown to outperform the previous non-constructive approaches, while learning an acceptable portion of the internal structure and bracketing rules as it predicts well-formed complex categories.

Prange et al. (2021) test the hypothesis that constructive supertagger models can generalise well over categories that are unseen in the training phase as they learn the internal structure of the supertags. Therefore, constructive tagging can eliminate the need to constrain the search space by using frequency-based cut-off thresholds to prune the supertag set, which is the strategy that most of the research that is discussed in *Section 3.5, p.28* implements. They build their method upon the notion that categories encode syntactic sub-trees and that they are converted to tree structures. The correspondence between syntactic categories that are encoded in lexical items of a CCG lexicon and phrase-structure rule definitions of CFPSG is demonstrated in *Section 2.3, p.7*, which can serve as an example to illustrate the underlying notion of their methodology.

Prange et al. (2021) first define a CFG that accepts the tokenised supertag strings to describe the recursive language of CCG syntactic categories, where non-terminals are composed of primitive categories and slashes that act as functors. Contrary to Bhargava and Penn (2020)'s flat linearised representation of category token prediction space, Prange et al. (2021) use a tree-structured RNN which predicts a tree in a top-down fashion. Given a token that is encoded with pre-trained word embeddings and prediction context, the model first predicts either a primitive category or a slash as the root node of the predicted tree. If the prediction for the root node is a functor slash, arguments of it are recursively predicted to add an additional depth to the tree. The base case is when all predicted child nodes are primitive categories. The results of Prange et al. (2021) provide evidence that constructive tagging using tree-structured predictions can recover majority of the long-tail categories that were previously discarded from the prediction space through cut-off strategies while obtaining state-of-the-art tagging accuracy.

CHAPTER 4

REPRESENTING OPEN-DOMAIN MEANING

This chapter introduces the Discourse Representation Theory (DRT; Kamp, 1981; Kamp and Reyle, 1993), which is a dynamic semantics theory that provides meaning representations with high-level expressivity. Contrary to first-order predicate logic (FOPL) semantics, which is used until this chapter to represent lexical meaning, DRT supplies a principled method to capture meaning beyond sentence boundaries. DRT and its extensions adequately represent certain phenomena observed in open-domain discourse and dialogue, such as referentiality introduced by indefinite noun phrases, tense, and presupposition.

At the beginning of this chapter, the limitations posed by FOPL that Montague Grammar uses in capturing open-domain meaning beyond the scope of a sentence are discussed. *Section 4.2* provides an informal definition for the minimal meaning-bearing unit of DRT, which is the Discourse Representation Structure (DRS), and demonstrate principles of translating natural language sentences to DRSs. *Section 4.3* and *Section 4.4* are on Segmented (S-DRT) and Projective (P-DRT) extensions of DRT, which respectively represent rhetorical relations and presuppositions and allowing linking anaphoric items to their referents across sentences based on the principle of accessibility. *Section 4.5* is about representing verb phrase meaning using neo-Davidsonian event semantics. *Section 4.6* illustrates how meaning representations of DRT are wrapped in λ-calculus expressions to account for the compositionality of interpretation. Finally, the chapter concludes by reviewing large-scale semantic treebanking efforts from the literature that use DRT with such extensions to annotate meaning of open-domain text.

Chapter 6 Cross-Level Typing the Logical Form uses the DRS language that is presented here to introduce the cross-level type assignment procedure on logical forms. Therefore, this chapter primarily serves as an introduction to the syntax of DRS language, and underlying linguistic motivations in adopting DRT to represent the meaning of multi-sentence open-domain text.

4.1 Meaning Beyond the Sentence Scope

FOPL meaning representations, such as the one used in *Section 2.4, p.10*, have limitations in representing the meaning of open-domain text and dialogues. FOPL represents the predicate-argument structure but only for sentence-level meaning.

Such semantic formalisms are partially inherited from Montague grammar (Montague & Thomason, 1975), which is based on predicate logic and λ-calculus. At the lexical-level, every constituent is assigned a corresponding logical form, which is either a constant or a function. Compositionality plays a central role in Montagovian grammar. The meaning of a sentence is assumed to be constructed as higher-order functions through function application of the meaning of its constituents. Syntax dictates the phrasal structure, and hence the order of constituent arguments, which undergo function application to derive sentence-level meaning.

Although, some lexical items, such as those that introduce coordination constructions and quantification, also contribute to the structure of the discourse beyond the sentence scope. Besides, inadequate representation of the rhetorical structure has immediate effects on resolving coreferentiality, which stems from how the semantic value of indefinite noun phrases is represented. The necessity for dynamic semantics theories, including DRT, to represent meaning beyond the scope of standalone sentences arises due to such observations.

For instance, the anaphoric nature of pronouns are examined with the **quantificational binding** approach, which was introduced by Reinhart (1976, 1981, 1983) and stems from the Government and Binding Theory (GBT; Chomsky, 1981). It suggests that a pronoun is bound by a quantifying expression within the sentence iff the expression precedes and c-commands the pronoun.

C-command is a syntactic constraint. A node in the syntax tree X **c-commands** another node Y iff X and Y do not dominate each other, and every node that dominates X also dominates Y. Node X **dominates** Y iff there is a bottom-up path from Y to X in the syntax tree. In other words, in a syntax tree, Y has to be, or to be contained by, a sister constituent of X for X to c-command Y.

(4.1) **Every runner** believes **he** is fast.

(4.2) [Every runner [believes [he [is fast]]]]

Consider the example in (4.1). Quantificational binding captures the semantic ambiguity posed by this sentence, as it permits two possible interpretations of it.

The first interpretation is the bound-variable reading. The pronoun '*he*' follows the universal quantifier '*every runner*', therefore satisfying the first constraint of quantificational binding. As seen in (4.2), the pronoun is also c-commanded by the quantifier; hence, we conclude in this reading that the pronoun is a variable that is bound by the universal quantifier. This analysis corresponds to the interpretation that *every runner believes that themselves are fast*.

The second interpretation is obtained by construing the pronoun '*he*' as a free variable that is not bound by the quantifier. This analysis yields the interpretation that *every runner believes that someone who is not themselves is fast*. Hence, the pronoun is considered corefential to an item that is not present in the context of the given sentence.

However, this approach surfaces inconsistencies in interpreting the infamous *donkey sentences* (Geach, 1962), an example of which is presented in (4.3).

(4.3) Every farmer who owns **a donkey**, beats **it**.

(4.4) [Every farmer [who [owns [a donkey]]] [beats [it]]]

(4.5) Every farmer who does not own a donkey, beats it.

Here we begin by hypothesising that the pronoun '*it*' is a variable that is bound by the existential quantifier, which is the indefinite noun phrase '*a donkey*'. The quantifier behinds the pronoun, so the first condition of bound-variable construal is met again. However, the c-command condition is not satisfied. As shown in (4.4), the quantifier is deeply embedded in the subordinate clause, thus it cannot c-command the pronoun. Hence, the bound-variable interpretation is not possible.

Similarly, we turn to the hypothesis of '*it*' being a free variable and coreferential. Nevertheless, this hypothesis does not hold as well. As illustrated in (4.5), if we negate the subordinate clause, the coreferentiality assumption between the pronoun and the existential quantifier breaks since the negated interpretation says that there is no donkey that is owned by each farmer to which the pronoun '*it*' can refer.

Similar problems are observed in the referentiality exhibited across sentences. Consider the example in (4.6). Pronouns '*she*' and '*it*' in (4.6)-b are respectively coreferential with the noun phrases '*Alice*' and '*a marathon*' from (4.6)-a. However, Montague grammar does not provide a principled method to capture the coreferents of such anaphora that have antecedents from a previous sentence.

(4.6) a. *Alice*$_1$ ran *a marathon*$_2$.
b. *She*$_1$ won *it*$_2$.

DRT overcomes these inconsistencies by assuming that each indefinite noun phrase introduces a referential item to a structured meaning representation. Moreover, it provides the mechanisms to merge individual sentence-level meaning representations to account for dynamically growing discourse.[1] This allows coreference relations to be resolved over structurally imposed accessibility constraints. The following section presents the basics of DRT over concrete examples.

4.2 Discourse Representation Theory

DRT is a **dynamic semantics** theory because it models mental representations of the hearer upon an utterance, which gets modified in time as the hearer obtains new information through unfolding the discourse (Kamp, 1981; Kamp & Reyle, 1993).

[1] The sentence comprehension models, such as the Garden Path model of Frazier (1979), can be discussed here together with the Late Closure and Minimal Attachment principles. Such models introduce principles to explain syntactic parsing and comprehension of standalone sentences over clausal attachment and parsimony constraints. Yet, DRT provides constraints at the discourse-level over the notion of accessibility, which helps to uniformly model compositional interpretation across analysis levels.

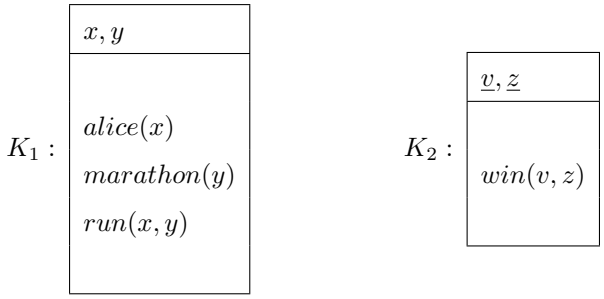


Figure 4.1: DRSs that represent the meaning of the sentences '*Alice ran a marathon*' and '*She won it*'.

DRS is the basic meaning-bearing unit in DRT. It is informally defined as follows. A DRS, K, consists of the set that captures the universe of discourse referents, U, and the set of discourse conditions, C, where $K = [\langle U, C \rangle]$.

Referents correspond to the referential items of the discourse, such as those that are introduced by noun phrases (e.g. $U = \{x, y\}$, where x corresponds to the indefinite noun phrase '*a farmer*', and y to '*a donkey*'). **Conditions** are either basic (e.g. $x = y$ representing coreferentiality between two referents), n-place predicates (e.g. $beat(x, y)$), or other DRSs.[2] Arguments of predicates are members of the set of discourse referents (e.g. $beat(x, y) \in C$, where $beat$ is the predicate and $x, y \in U$).

The sentence from (4.6)-a is used to provide a running example. The corresponding DRS for this sentence, K_1, is presented in Figure 4.1. Here, the (:) notation is used to assign a label to each DRS.

K_1 has two referents $\{x, y\}$, where the definite noun phrase '*Alice*' introduces x to the discourse and the indefinite noun phrase '*a marathon*' introduces y. The set of discourse conditions contains three elements $\{alice(x),\ marathon(y),\ run(x, y)\}$. They represent the predicate-argument structure. The variable x is an argument of the predicate '*alice*', y is the argument of '*marathon*', and the last condition $run(x, y)$ denotes that the agent of the predicate run is the referent which corresponds to Alice, and the theme is the one that corresponds to a marathon.

Similarly, the sentence (4.6)-b that follows in discourse translates to K_2 as shown in Figure 4.1. This time noun phrases that are in subject and object position are pronouns. The anaphoric nature of pronouns is reflected in K_2 by introducing referents v and z that are not associated with a referential item from the discourse[3]. The referents v and z only appear as an argument of the predicate win of the single available condition.

[2] Henceforth, the term *predicate* is used to refer to the logical predicates that appear in interpretation or predicate-argument structure. The syntactic predicate, in the context of a lexicalised grammar theory such as CCG, refers to a constituent that is the functor that combines with an adjacent argument constituent. The logical predicate that represents the meaning of the constituent might diverge from the surface form. An example of this distinction is apparent in the representation of idioms (Bozşahin & Güven, 2018). Consider the verbal constituent '*spill the beans*', which is a syntactic predicate on the surface, whereas its associated interpretation might represent the meaning of this constituent with the logical predicate '*reveal*', as in $\lambda x.reveal' x$.

[3] By convention, unbound referents are emphasised with underline.

Note that Figure 4.1 uses **box-notation**.[4] It is a graphical representation that elucidates the referent and condition sets and their relations to the predicate-argument structure in a compartmentalised box. Alternatively, DRSs can also be represented in flat notation, as in (4.7). Box and flat notations are equivalent.

(4.7) $K_1 : [\langle\{x, y\}, \{alice(x), marathon(y), run(x, y)\}\rangle]$
$K_2 : [\langle\{v, z\}, \{won(v, z)\}\rangle]$

A DRS is translated to FOPL with a straightforward procedure. The translation function is defined in (4.8) (Blackburn & Bos, 2005). This function maps every discourse referent to an existential quantifier in FOLP and translates the predicate-argument structure recursively. For instance, the above DRSs are mapped to FOPL expressions that are in (4.9).

(4.8) $[\langle\{r_1, ...r_n\}, \{c_1, ..., c_n\}\rangle]^{fol} \stackrel{def}{=} \exists r_1 ... \exists r_2 (c_1 \wedge ... \wedge c_n)$

(4.9) $K_1 : \exists x \exists y (alice(x) \wedge marathon(y) \wedge run(x, y))$
$K_2 : \exists v \exists z (win(v, z))$

DRSs are combined to capture the meaning of discourse that unfolds across sentences. That is done via the *MERGE* function ($\oplus$), which is defined in (4.10). It is a function of arity 2, whose domain is defined over DRSs.

(4.10) $MERGE(K, K') = [\langle\{U_K \cup U_{K'}\}, \{C_K \cup C_{K'}\}\rangle]$

A merged DRS is obtained by taking the union of referent and condition sets of each DRS. Figure 4.2 shows the resulting DRS in box-notation after merging the DRSs from Figure 4.1.

To show the case where a DRS might have complex conditions, consider K_1 from Figure 4.3. This DRS corresponds to the negated form of the previous example, '*Alice did not run a marathon*'. Notice that this time scope of negation is captured with an embedded DRS, K_2, that is marked with negation scope, $\neg$. The outermost DRS, K_1, which is created after introducing the sentence to an empty context, only contains x as a referent which corresponds to '*Alice*' as captured with the condition $alice(x)$. Contrary to the first example, this time the referent y, which is introduced by the indefinite noun phrase, now belongs to the negated DRS, K_2.

The implication of representing negation scope like this becomes clear when we merge the DRS of this sentence with the representation of the follow-up sentence. As seen

[4] There are also alternative notations to represent DRSs such as the clausal-form notation or the linearised form, which are commonly employed in the processing of the DRSs in computational modelling. Among those alternative notations, the former enables efficient similarity matching of two given DRSs. The latter is frequently employed as the input and output representation under the sequence-to-sequence modelling paradigm. A definition of these notations are provided in *Chapter 6 Cross-Level Typing the Logical Form* together with an illustration of their usage in DRS-based semantic parsing models.

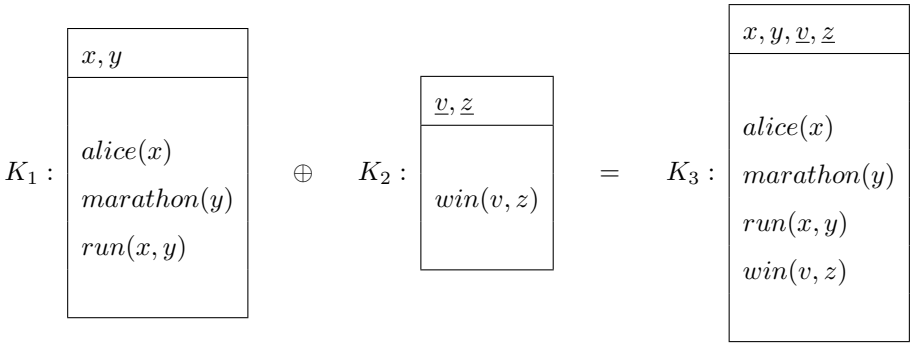


Figure 4.2: Merge of two DRSs that represent the meaning of the sentences '*Alice ran a marathon*' and '*She won it*'.

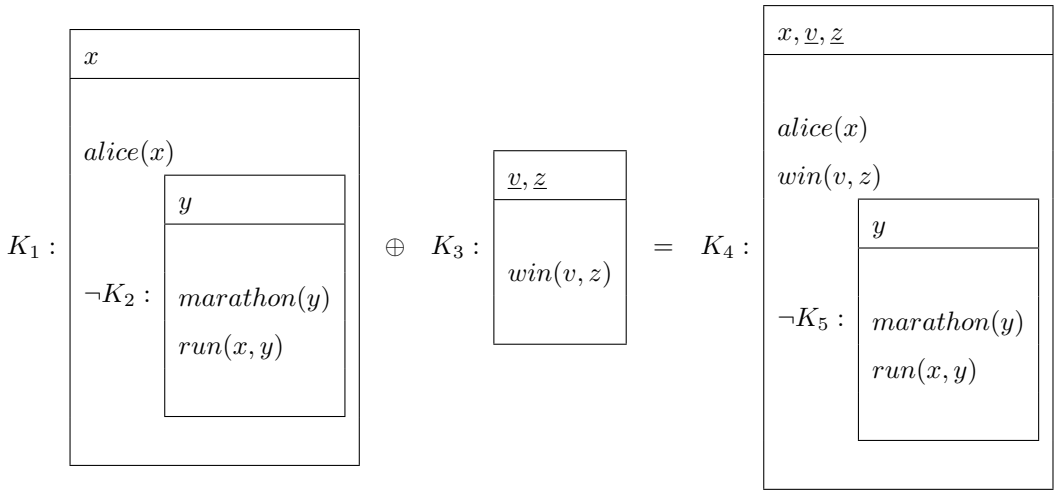


Figure 4.3: Scope of negation in DRT. Merge of two DRSs that represent the meaning of the sentences '*Alice did not run a marathon*' and '*She won it*'.

from Figure 4.3, in merged DRS, K_4, the under-specified referents v and z do not have access to the scope of the negated DRS condition.

The structure of the DRS imposes **accessibility relations** on referents, which help us link coreferential items. Some accessibility relations are presented in (4.11) (Geurts et al., 2020). Every DRS is accessible to itself. Let K' and K'' be DRSs, if another DRS, K, has a condition of the form:

(4.11) a. $\neg K'$, then K is accessible to K'
b. $K' \vee K''$, then K is accessible to K' and K''
c. $K' \Rightarrow K''$, then K is accessible to K' and K' is accessible to K''

In other words, DRSs that are accessible to a given DRS, K, are found by tracing the DRSs in left and upwards direction starting from K. The negated example in

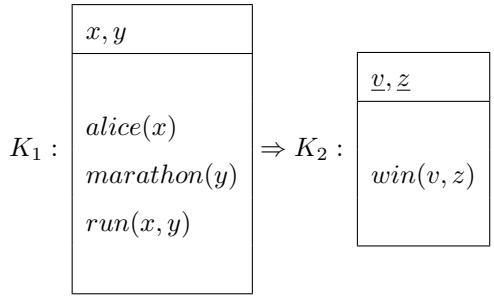


Figure 4.4: Scope of implication in DRT. DRS that represents the meaning of the sentence '*If Alice runs a marathon, she wins it*'.

Figure 4.3 is of the form (4.11)-a; therefore, K_5 can access K_4 but not vice versa. Likewise, Figure 4.4 is an example of the DRS condition of the form (4.11)-c, where K_1 is accessible to the implied DRS K_2.

Given the above accessibility relations, the **accessibility constraint** on pronouns for anaphora resolution is defined as follows (Geurts et al., 2020):

> Discourse referent x that is introduced to DRS K' by a pronoun, is equated to another referent y iff y belongs to a DRS K that is accessible to K'.

To illustrate, in Figure 4.2, the accessibility domain of K_3 is the set of referents $\{x, y\}$. Thus, under-specified referents v and z that belong to K_3 are respectively equated to x (she = Alice) and y (it = marathon). This is reflected in Figure 4.5-a with additional discourse conditions.

The scope of logical implication has similar accessibility relations. In Figure 4.4, K_1 is accessible to the implied DRS K_2; therefore, referents v and z are equated to their antecedents as in Figure 4.5-b.

The accessibility domain is constrained in the case of negation scope. For K_4 in Figure 4.3, the set of accessible referents that are linked to the anaphoric referents is $\{x\}$. In Figure 4.5-c, we equate v to its possible antecedent x, though referent z (corresponding to the '*it*' pronoun) cannot be equated to y, since K_5, to which y belongs, is not accessible to K_4.

DRSs are model-theoretic interpretations whose truth values are verified with respect to a given first-order model $M = \langle D, I \rangle$, where D is a non-empty set of relations, and I is the set of constants of the model that correspond to the discourse referents. DRT uses **embeddings** (or assignments), which are partial functions from the discourse referent domain into the range D. The truth value of a DRS is verified by finding an embedding from M whose domain includes the referents of the given DRS. Hence, every DRS allows automated semantic interpretation given a model (Muskens, 1996).

4.3 Segmented DRT

S-DRT extends the scope of representation to capture rhetorical relations and trace anaphora resolution in discourse (Asher, 1993; Asher & Lascarides, 2003; Lascarides & Asher, 2008). As shown in *Section 4.2, p.33*, standard DRT provides accessibility conditions to capture anaphora resolution beyond the scope of a single sentence. However, it falls short of accounting for the resolution constraints imposed by the discourse structure.

Consider below multi-sentence example:

(4.12) a. Alice run a marathon.
b. She trained for it.
c. She ate healthy meals.
d. She won it.

We get the corresponding DRS for each sentence using the described translation method as presented in Figure 4.6. Notice that the merge yields a representation, where the referent n, which is introduced by the pronoun '*it*' in (4.12)-d, is potentially equated to an accessible referent from the set $\{y, l\}$, where y is introduced by the indefinite noun phrase '*a marathon*' in (4.12)-a, and l by '*healthy meals*' from (4.12)-c. Although, common sense construal only picks the referent y ('*a marathon*') as a coreferent for n.

Hobbs (1985) and Hobbs et al. (1993) discuss such observations, which lay the ground for S-DRT as it is formalised in Lascarides and Asher (2008). An informal defi-

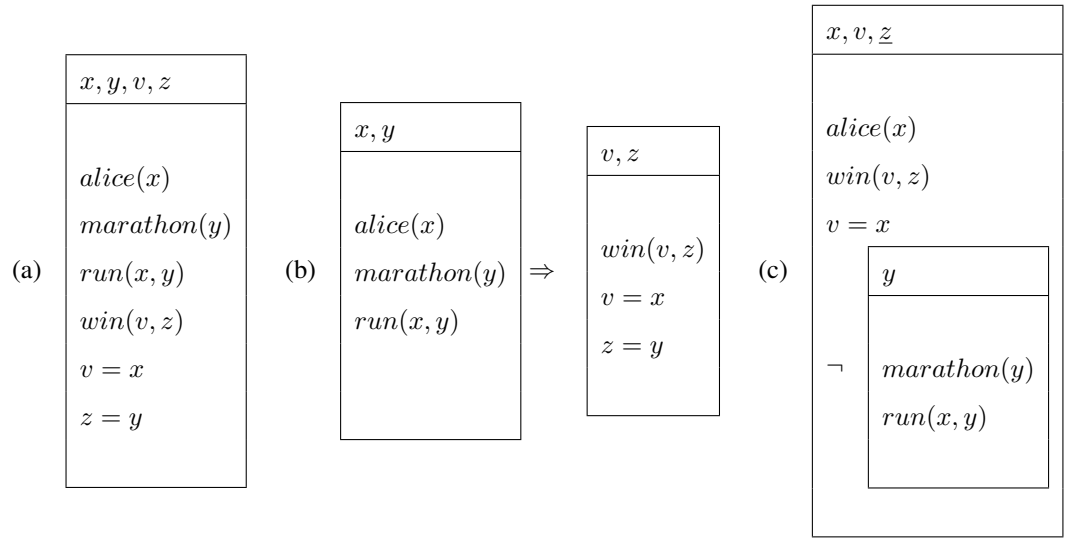


Figure 4.5: Anaphora resolution in DRT. Illustration of (a) equating referents introduced by pronouns to their antecedents, (b) accessibility domain in logical implication, (c) constrained accessibility domain due to the negation scope.

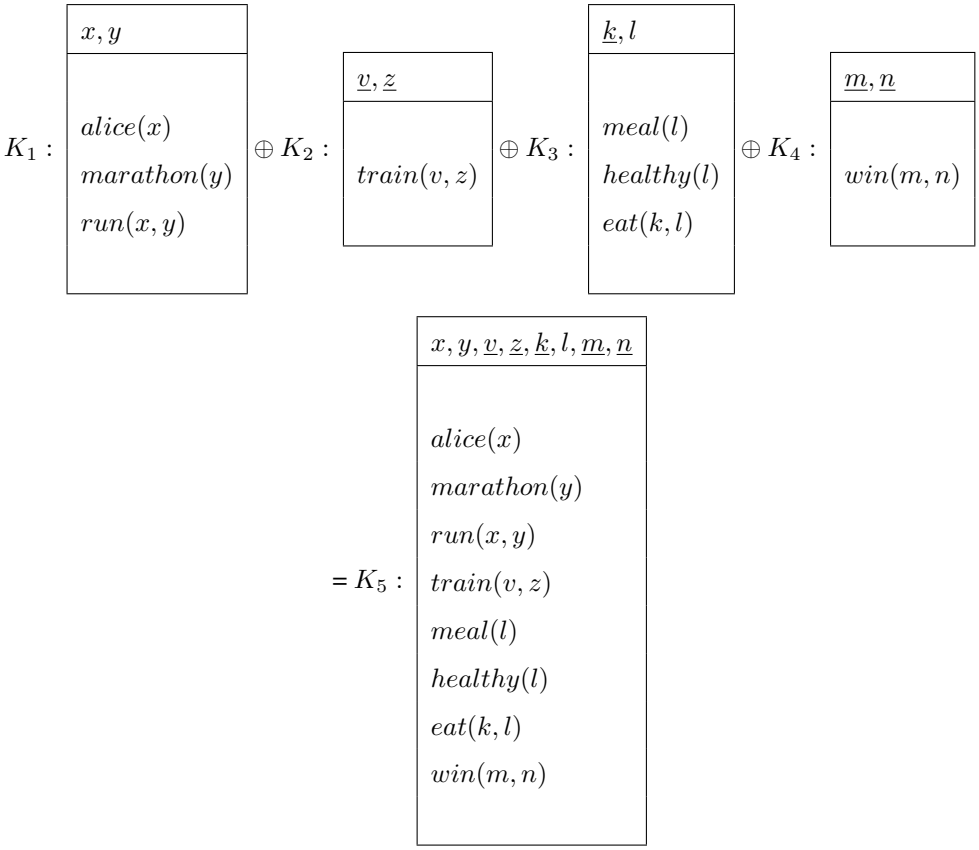


Figure 4.6: Representation of the discourse structure with standard DRT.

nition of S-DRT starts with assuming a set of **discourse relations**, R (e.g. $R = \{Continuation, Contrast, Elaboration, ..., Narration, Parallel\}$). Discourse relations are either subordinating or coordinating (Polanyi, 1985). Subordinating relations partition the discourse structure vertically, whereas coordination relations introduce horizontal partitioning. For instance, *Elaboration* is subordinating, and *Narration* is a coordinating relation.

Discourse structure is assumed to be assembled with respect to a constraint known as the **Right-Frontier Constraint** (Polanyi, 1985; Webber, 1988). Given a new sentence (or clause), the anaphora from this sentence (or clause) is only bound to an antecedent on the right-frontier of the discourse, which is the latest bound right-most node, or a node that dominates it.

Figure 4.7 presents discourse structure as it gradually unfolds with sentences from (4.12). At each step, the right-frontier nodes are displayed with dashed frames. In Figure 4.7-b, the binding of clause π_2 to the antecedent introduces a vertical partitioning to the discourse where the relation between (4.12)-a and (4.12)-b is captured as *Elaboration*. Likewise, clauses π_3 and π_4 are bound to their antecedents with the coordinating *Narration* discourse relation, introducing horizontal partitioning.

In Figure 4.7-b and Figure 4.7-c, the discourse right-frontier consists of two nodes to which π_3 and π_4 can bind. Asher and Lascarides (2003) and Lascarides and

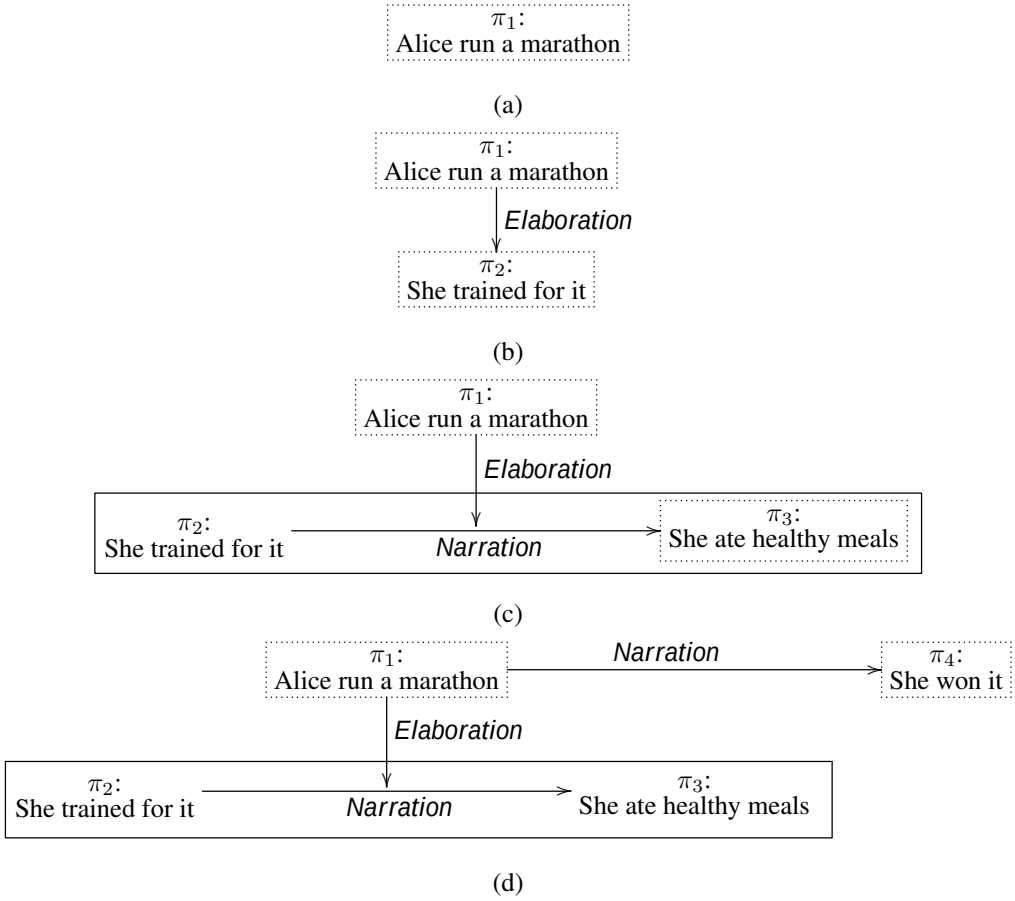


Figure 4.7: Discourse structure unfolding through the binding of clauses to the right-frontier.

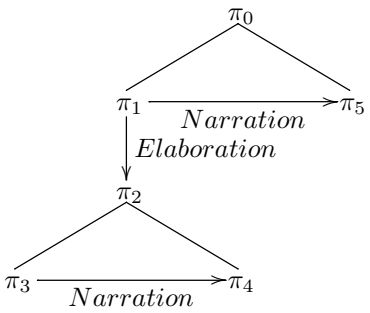


Figure 4.8: Hyper-graph representation of the discourse structure for a multi-sentence paragraph.

Asher (2008) use a heuristic, **Maximal Discourse Coherence** (MDC), to rank the interpretations when there is such ambiguity. MDC prioritises maximisation of (i) number of rhetorical connections, (ii) number of anaphora which is resolved to a coreferent, and (iii) quality of rhetorical relations after binding the current clause to

discourse structure.

Following Lascarides and Asher (2008), in S-DRT an S-DRS, which represents the discourse structure, is defined as a 3-tuple $\langle A, LAST, F\rangle$, where A is a set of speech act discourse referents (e.g. labels that are assigned to the representation of each clause), $LAST$ is a variable that points to the last clause that is introduced to the discourse, and F is a function from domain A into a range of S-DRS.

In this context, the above discourse structure is represented by the hyper-graph from Figure 4.8. Here, the speech act discourse referent set is $A = \{\pi_0, ..., \pi_5\}$. The last clause bound to the discourse is $LAST = \pi_5$, which determines the right-frontier. The mapping between discourse referents and the corresponding S-DRS formula is as follows:

(4.13) $F(\pi_0) = Elaboration(\pi_1, \pi_2) \wedge Narration(\pi_1, \pi_5)$
$F(\pi_1) = K_1$
$F(\pi_2) = Narration(\pi_3, \pi_4)$
$F(\pi_3) = K_2$
$F(\pi_4) = K_3$
$F(\pi_5) = K_4$

The equivalent box-notation for this S-DRS is in Figure 4.9-a. Notice that, distinctive from the standard DRT meaning representations, an S-DRS now reflects the discourse structure. An immediate consequence of this is an update on the accessibility relations notion. With S-DRT, the right-frontier constraint is inherently encoded in the structure of an S-DRS; therefore, anaphora resolution is guided by the discourse structure.

For example, in Figure 4.9-b, the referent n of π_5 (which is introduced by the pronoun '*it*' of (4.12)-d) has access to only the set of referents $\{x, y\}$, which helps us resolve its coreferent as y (which is introduced by the indefinite noun phrase '*a marathon*' of (4.12)-a).

4.4 Projective DRT

A presupposition is an implicitly assumed knowledge or belief whose truth value is presumed in a given discourse. Certain linguistic constructions and constituents introduce presupposition, and in certain languages, such as English, it is a lexical phenomenon. Some examples which introduce presupposed knowledge in a discourse context are definite noun phrases (e.g. (4.14)-a; presupposed knowledge of a specific pair of shoes existing), factive or implicative verbs (e.g. (4.14)-b; presumed knowledge of the fact that Alice was the person who completed the action of buying that specific pair of shoes), it-clefts (e.g. (4.14)-b; similar to the factive verb example) (Geurts et al., 2020).

(4.14) a. Alice bought **the shoes**.
b. John **realised that** Alice bought the shoes.

c. **It is the case that** Alice bought the shoes.

d. If Alice bought the shoes, she is ready to run.

So far, *Section 4.2, p.33* have shown how standard DRT captures proper nouns and definite noun phrases as referents, which are referenced as anaphora of antecedents under accessibility conditions. Proper nouns and definite noun phrases are also known to trigger presuppositions. Although, presupposition and anaphora are two very similar phenomena, the former has peculiar characteristics.

Presuppositions escape the embedded clausal scopes under negation, modal operations, and implication due to their existential indifference to such operators (Bos, 2003). This is known as **projective behaviour**, and it is differentiated from asserted information

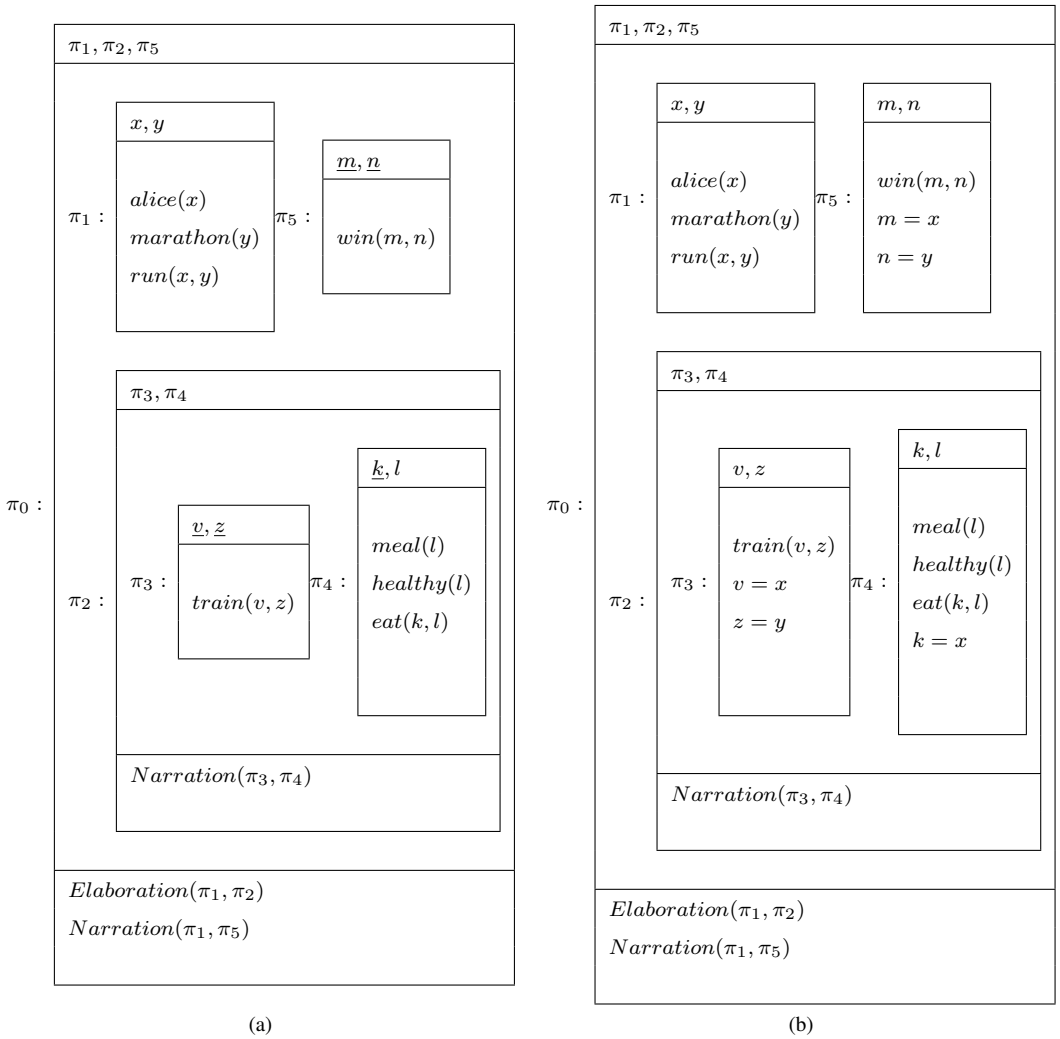


Figure 4.9: Representation of the discourse structure with S-DRT. Illustration of (a) S-DRS that represents the meaning of multi-sentence paragraph, (b) This S-DRS after anaphora resolution.

(Venhuizen, Bos, et al., 2013). For instance, (4.14)-d is the same sentence as (4.14)-a, but this time in a conditional construction. Here the definite determiner '*the*' which modifies the noun '*shoes*', and the proper noun '*Alice*', respectively introduces projected information of a pair shoes, and a person named Alice to exist, which are part of the assertion that the person accomplishes the action of acquiring the possession of those pair of shoes.

Van der Sandt (1992) presents empirical evidence that supports the hypothesis that projected content behaves similar to anaphora in the sense that it might bind to an accessible antecedent. This hypothesis is also implemented in early DRS parsers, such as Boxer (Bos, 2008), that generate DRSs that mark the anaphoric nature of presuppositions. However, different from anaphora, the antecedent of the presuppositions might not always be found in the discourse context. Under DRT, this behaviour brings up the necessity to introduce a resolution stage where ad-hoc antecedents are added at an accessible level of discourse to which the projected material can bind. This problem is known as the **accommodation of the presupposition**.

One approach that accounts for the accommodation problem is the Layered DRT (L-DRT; Geurts and Maier, 2013), which marks the type of the information content per discourse referent and condition to trace whether they are part of the projected, asserted, or conventional implication layers. However, this approach does not impose any constraints on the special layers that are incorporated into the representation and introduces additional complexity by assuming special treatment of each layer in terms of binding and accessibility.

A computationally sound alternative to L-DRT is the Projective DRT (P-DRT; Venhuizen, Bos, et al., 2013), which is an extension to DRT that formalises the binding of presuppositions to antecedents as a problem of pointer assignment. The formalism is implemented as a deterministic parser for English (Venhuizen & Brouwer, 2014).

The underlying assumption of P-DRT is to assign labels to Projected DRSs (P-DRS) and pointers to the referents and conditions of each P-DRS. The corresponding P-DRS for the sentence in (4.14)-d is shown in Figure 4.10. The asserted material is always interpreted at the site where it is introduced, meaning that the referent and conditions that correspond to the asserted information point to the label of the P-DRS they belong to (e.g. $buy(x, y)$ as an asserted material has a pointer to its introduction site, which is the P-DRS that has the label P_2). On the contrary, projected material can point to any accessible P-DRS or it is a free variable (e.g. referents x and y, conditions $alice(x)$ and $shoes(y)$ have assigned free variable pointers since they are projected material). The pointer assignment mechanism introduces a clearly traceable distinction between projected and asserted content while eliminating the need for an additional presupposition binding resolution step.

4.5 Neo-Davidsonian Event Semantics

Traditional representations of predicate-argument structure in FOPL cannot adequately capture certain aspects of verb semantics (such as thematic role and tense) and verb phrase modifiers.

Thematic role is the assigned semantic role of a noun phrase in relation to a verb that governs the noun phrase. For example, consider the main transitive verb of (4.15)-a, '*ran*'. The subject noun phrase '*Alice*' bears the *Agent* role (i.e. an individual performing the action denoted by the verb), and the object '*a marathon*' has the role *Theme* (i.e. an individual that undergoes the action while not changing its state). In traditional FOPL representation, semantic roles are implicitly encoded in the order of arguments that the predicate takes (e.g. the first argument being the *Agent* and the second *Theme*).

Such an implicit representation also causes problems when representing the meaning of verb phrase modifiers. For example, in (4.15)-b, the adverb '*fast*' has semantics that modifies the verb '*ran*', but this relation cannot be represented. As illustrated, the best attempt is to introduce an unary predicate for the adverb that takes the verbal predicate as an argument.

(4.15) a. Alice ran a marathon.
$\exists x \exists y(alice(x) \wedge marathon(y) \wedge run(x, y))$

b. Alice ran a marathon fast.
$\exists x \exists y(alice(x) \wedge marathon(y) \wedge fast(run(x, y)))$

Considering these limitations, Davidson (1967) proposes reifying events by introducing an event variable to the representation, as in (4.16)-a. Events are commonly represented with the variable *e* that indicates an event-like individual in the first-order model. This is similar to typing variables of the model to distinguish between entity and event-like individuals.

(4.16) a. $\exists x \exists y \exists e(alice(x) \wedge marathon(y) \wedge run(e, x, y))$

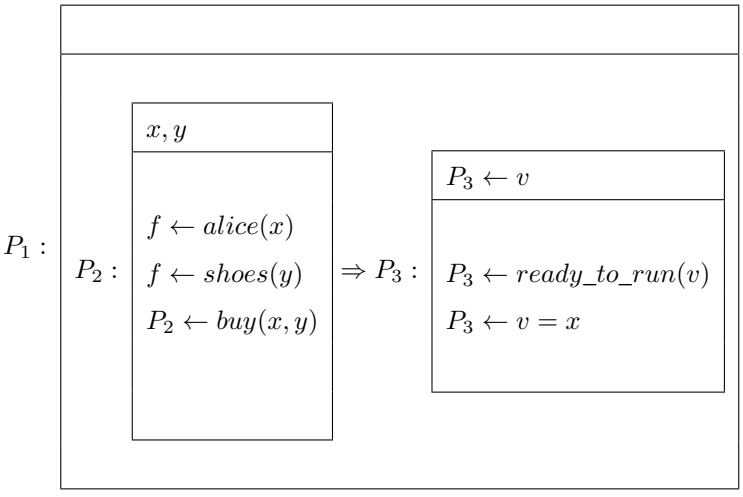


Figure 4.10: P-DRS that represent the meaning of the sentence '*If Alice bought the shoes, she is ready to run*' by differentiating the asserted and projected behaviour with DRS labels, and referent and condition pointers.

b. $\exists x \exists y \exists e(alice(x) \wedge marathon(y) \wedge run(e, x, y) \wedge fast(e))$

With the Davidsonian approach, the latter problem regarding the representation of verb phrase modifiers is solved. As in (4.16)-b, adverbs introduce predicates that have the event variable introduced by the modified verb as an argument.

Parsons (1980, 1990) proposes an alternative where the thematic relations of event meaning are captured over binary predicates, whose arguments are an event and an entity variable. This approach is known as the **neo-Davidsonian event semantics** representation in the literature (Bos, 2008, 2009a, 2009b; Sayeed & Demberg, 2012). The neo-Davidsonian representation for the above sentences is in (4.17).

(4.17) a. $\exists x \exists y \exists e(alice(x) \wedge marathon(y) \wedge run(e) \wedge agent(e, x) \wedge theme(e, y))$

b. $\exists x \exists y \exists e(alice(x) \wedge marathon(y) \wedge run(e) \wedge agent(e, x) \wedge theme(e, y) \wedge fast(e))$

Notice that in (4.17)-a the semantic roles of the subject and object are made explicit with standalone predicates that bind an entity to an event variable over a thematic role (e.g. *agent(e, x)* denotes that the individual *x* is assigned the *Agent* semantic role by the event *e*). This helps to keep event predicates unary (e.g. *run(e)*), so that variation in intransitive, transitive and ditransitive use of certain verbs are handled regularly while arguments are offloaded to separate predicates that bind the semantic role of each argument (Abzianidze & Bos, 2019). The optional predicate for verb phrase modifiers is inherited from Davidsonian representation.

This approach is further expanded to account for the sub-lexical morpho-syntactic phenomena such as tense. The sub-categorisation of existentially quantified variables is enlarged to include temporality variables, which are commonly represented with *t*. As shown in (4.18)-a, given the temporality variable, the tense of the verb phrase is captured with a unary predicate that ranges over event variables (e.g. *past(e)*).

(4.18) a. $\exists x \exists y \exists e \exists t(alice(x) \wedge marathon(y) \wedge run(e) \wedge agent(e, x) \wedge theme(e, y) \wedge past(e)$

b. $\exists x \exists y \exists e \exists t(alice(x) \wedge marathon(y) \wedge run(e) \wedge agent(e, x) \wedge theme(e, y) \wedge in(e, t) \wedge precedes(t, now))$

An alternative but more elaborate approach to marking tense is to use a logical constant that anchors to the present and to represent the relation of the verb tense with respect to this anchorage, as in (4.18)-b (e.g. *in(e, t)* $\wedge$ *precedes(t, now)*).

Section 4.2, p.33 shows that DRSs can be translated to FOPL expressions. Then, the above neo-Davidsonian event semantics is accommodated in DRT. This is achieved by introducing event and temporality variables as discourse referents and predicates as discourse conditions in DRS. For instance, Figure 4.11 shows the corresponding DRS with neo-Davidsonian event semantics for the FOPL expression from (4.18)-b.

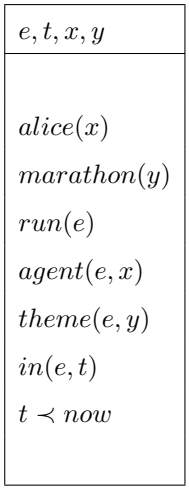


Figure 4.11: DRS with Neo-Davidsonian event semantics that explicitly capture thematic role and tense.

4.6 Compositional Interpretation with DRT

The DRS translations for sentence-level meaning are provided in the previous sections of this chapter. This conceals the underlying compositional process of deriving interpretations from lexical meaning. Similar to the syntactic process that is described in *Section 2.4, p.10*, lexical meaning is represented with DRT using function abstraction and application of λ-calculus together with CCG's categorial transparency principle.

Bos (2003, 2008, 2009b) introduces the concept of partial DRS to represent lexical meaning within DRT. A **partial DRS** is an under-specified meaning representation assigned to each syntactic category at the lexical-level. They are λ-abstractions, and similar to Montague grammar, the meaning of a sentence is derived through function application of the meaning of its constituents, which are λ-DRSs. Overall, λ-abstracted lexical DRS semantics is the glue language for compositional interpretation.

In this setting, the skeletal semantic structure of lexical constituents of English that have the syntactic category `N` (noun) and `NP/N` (determiner) is presented in Figure 4.12-a. Here, $pred$ is a placeholder for free-form predicates that appear in DRS conditions, dependant on the lexical item. Consider the toy CCG lexicon from Figure 4.12-b that is composed of 2 entries, where $pred$ is realised as '*marathon*' in the DRS condition of noun item.

Such under-specified semantic values assigned to each lexical constituent in lexical analysis are reduced through derivation with combinatory rules. The process involves a series of function application, β-reduction, α-conversion of bound variables, and DRS merge operations. Figure 4.13 illustrates the interpretation for the quantified noun phrase '*the marathon*' which is built bottom-up from the lexical meaning of '*the*' and '*marathon*' using forward application as defined in (2.12) to yield a constituent with category `NP`. The resulting logical form is expanded to show each step in the semantic application.

Figure 4.14 presents the compositional interpretation of a standalone sentence. In this example, DRS referents are sub-categorised and represented with distinct symbols (e.g. v for lambda, x for entity, e for event, and t for temporal variables). Each referent from the corresponding category is distinguished with an index (e.g. v_1 and v_2 for bound λ-variables, and x_1 for the first entity variable within a DRS scope). Thus, α-conversion is regarded as re-indexing variables of each sub-category to avoid name-collusion before reduction.

Note that, unlike the FOPL logical form expressions that are presented in *Section 2.4, p.10*, with partial DRS semantics, the arity of lexical logical forms does not match the arity of the assigned lexical syntactic category. For example, the transitive verb '*ran*' with function category $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}$ of arity 2 has a logical form with 3 λ-abstractions. Bos (2009b) names this approach as the **method of continuation**.

Specifically, the semantics of the S (sentence) category is a λ-abstraction itself (see the last step of derivation in Figure 4.14) to account for further possible combination with a sentence modifier (e.g. constituents with category $\mathsf{S}|\mathsf{S}$). This approach introduces event variables to the interpretation as part of the DRS that belongs to the lexical semantics of verbs. Hence, it captures the correct scope for sentences with negation, quantification, or implication semantics while allowing for compositional construction of DRS representations beyond single sentences.

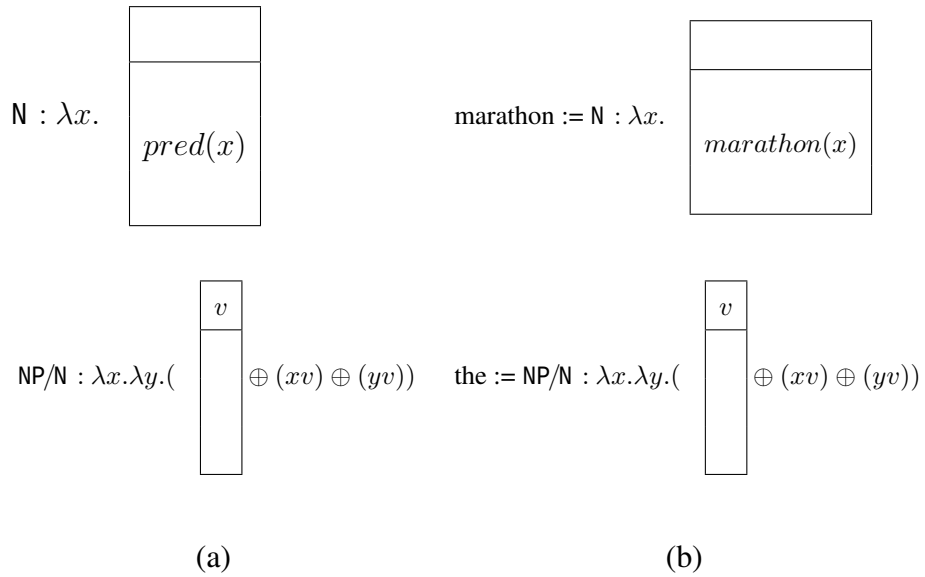


Figure 4.12: Skeletal semantic structure of partial DRSs. Illustration of (a) Partial DRSs for English nouns and determiners, (b) A Toy CCG lexicon with items that have partial DRS lexical semantics.

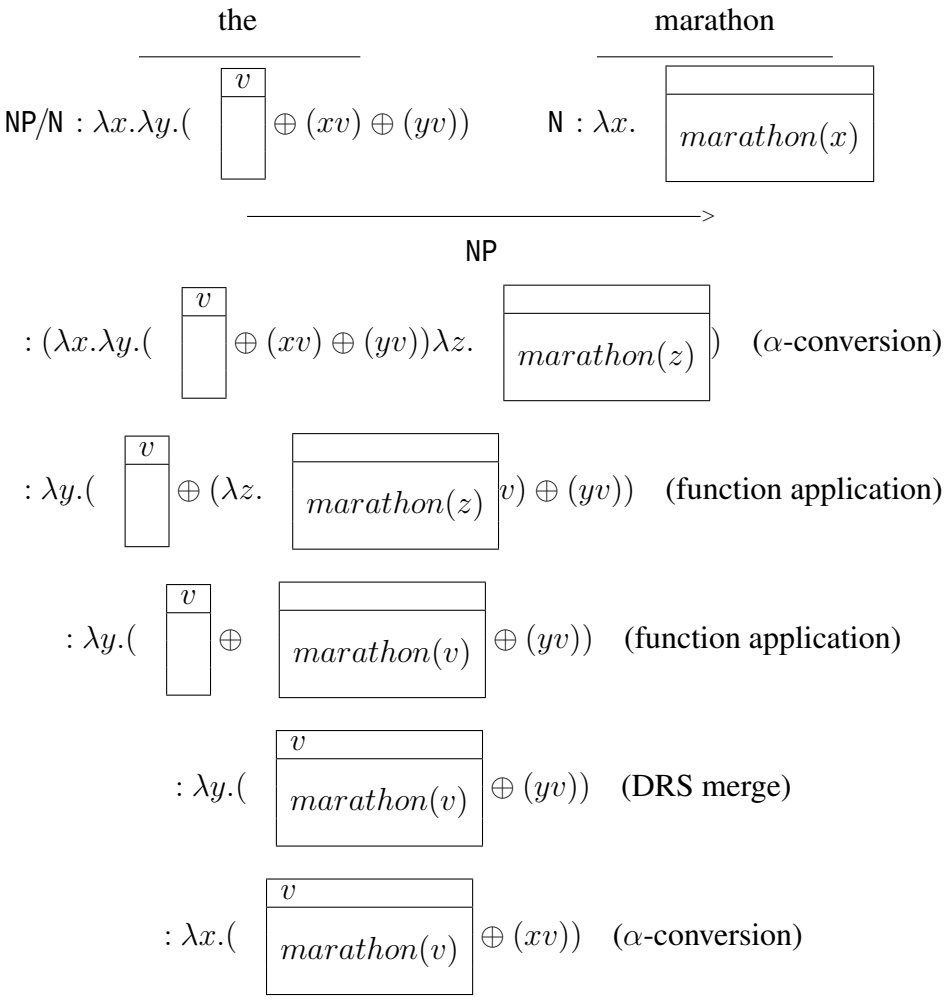


Figure 4.13: Deriving the compositional interpretation of the English quantified noun phrase '*the marathon*'.

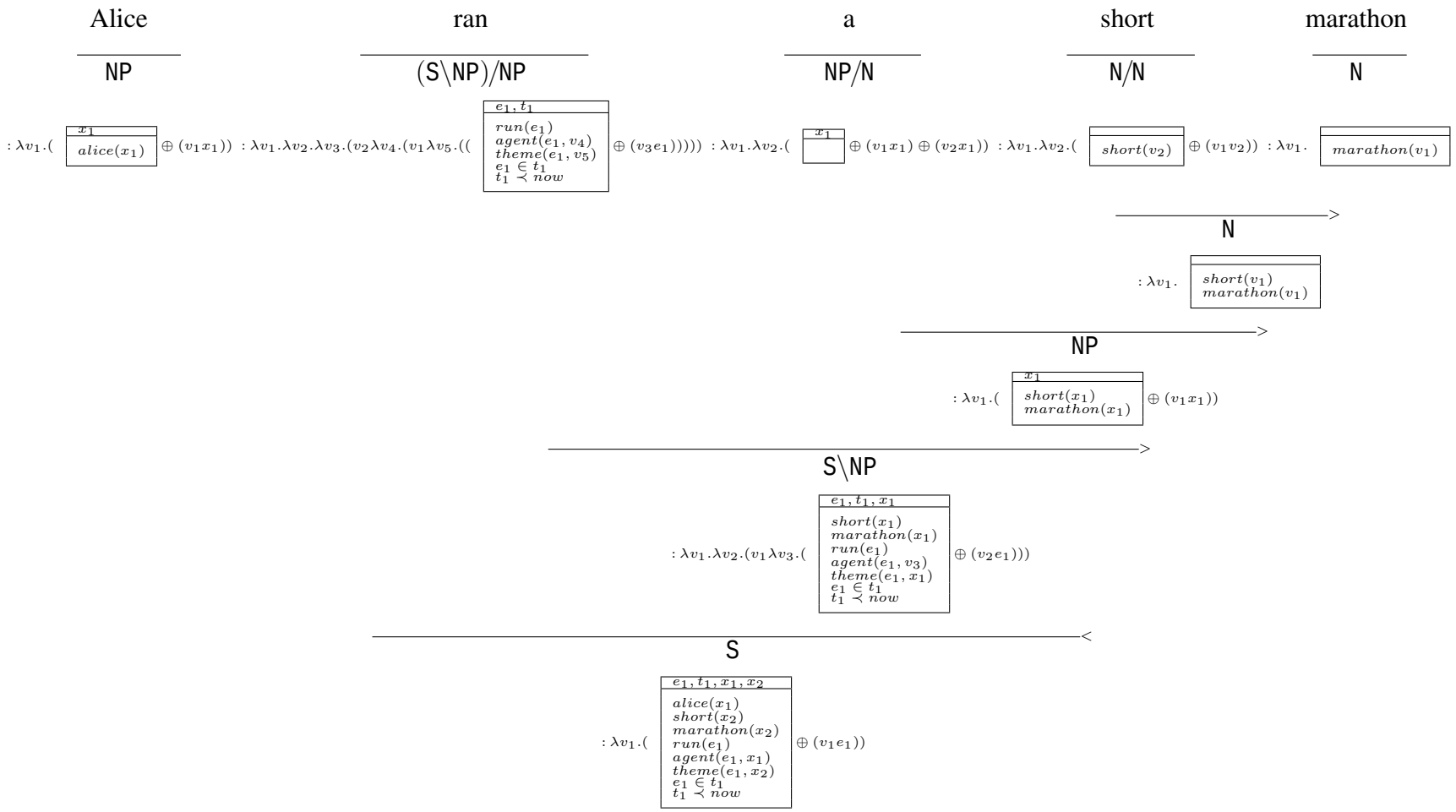


Figure 4.14: Bottom-up construction of the DRS interpretation for the sentence '*Alice ran a short marathon*'.

4.7 Semantic Treebanking with DRT

A semantic treebank (henceforth SemBank) annotates the underlying semantic structure of the text with varying degrees of granularity together with the morpho-syntactic features. Semantic features that are used in SemBanking can span the phenomena that occur at lexical-level up to the level of discourse structure. Shallow semantic annotations that mostly labels phenomena that are observed at lexical-level consist of word senses, named entity classes, thematic roles of verbs and prepositions, and the animacy of sentence-level predicates.

Certain SemBanks also annotate the text with deep semantic annotations that represent the meaning of the sentence or the document in hand using expressive meaning representations that are loosely or strictly translatable to first-order logic (Bos, 2016). Recent examples of such meaning representations that are used in SemBanking literature are Abstract Meaning Representations (AMR) and DRT. An example resource that annotates logical forms of sentences using the former formalism is the AMR bank (Banarescu et al., 2013).

SemBanks, like treebanks, are used to add supervision to the learning process of computational models of semantic analysers, which map natural language text to its meaning. *Chapter 5 Semantic Analysis* introduce a subset of those models from the literature. These models use DRT as the representation of the target prediction space and utilise two recently developed large coverage DRT SemBanks to supervise model learning: Groningen Meaning Bank (GMB; Basile et al., 2012a; Bos et al., 2017) and Parallel Meaning Bank (PMB; Abzianidze et al., 2017).

GMB is a monolingual resource that annotates English documents. Each GMB document consists of multiple sentences. The meaning of each sentence that makes up a document and the document-level meaning that is compositionally constructed from the meaning of individual sentences are annotated as a DRS with Segmented and Projective extensions, and neo-Davidsonian event semantics representation. The underlying corpora of GMB consists of documents from 5 domains. These genres cover fictional text in the form of jokes and fables; news documents (sampled from Voice of America; Heil, 2003); factual descriptive text (sampled from The World Factbook; Geck, 2017); and open-domain written and transcribed spoken form text that represents contemporary American English (sampled from the MASC corpus of Ide et al., 2010 that is composed of sentences from the Open American National Corpus; Ide and Suderman, 2004). The resource is made up of 10,000 documents that are composed of 62,010 individual sentences.

PMB is a multilingual resource that only annotates sentence-level meaning in four languages. Annotated sentences are parallel translations in English, German, Italian, and Dutch. The corpora that PMB annotates hold greater variation in covered genres compared to GMB. In addition to sentences from news documents, text from datasets that represent textual entailment (Giampiccolo et al., 2007), fictional text from English literature and religious texts such as the Bible (Christodouloupoulos & Steedman, 2015) are also used in sampling sentences of PMB. The English portion of PMB is composed of 285,154 sentences. Not all English sentences are translated to and annotated in other languages of PMB in the latest release of the resource. However, the

SemBank is dynamically growing with the aim of parallelising all English sentences with language-specific syntactic and semantic annotations on all non-English languages it covers.

GMB follows a multi-layered annotation scheme. It provides syntactic and semantic annotations at the lexical, sentence, and full document levels. The annotations are first bootstrapped using computational models that are trained on representative datasets. The bootstrapping stage generates semi-gold annotations, which does not guarantee the correctness of the annotations. In follow-up stages, a subset of annotations are checked by human experts and corrected if necessary as part of a human-in-the-loop annotation process using gamification methods (Basile et al., 2012b; Venhuizen, Basile, et al., 2013).

In order to bootstrap annotations on GMB corpora, the documents are initially split into individual sentences, and sentences are tokenised using a statistical sentence-boundary detector and tokeniser (Evang et al., 2013). To represent the syntactic structure of individual sentences, POS annotations are provided for the tokens using the PTB POS label set with minor deviations. Given the POS labels, CCG categories are inferred from a closed set of all possible syntactic categories that can parse open-domain English sentences. The part-of-speech labels and syntactic categories are bootstrapped using the predictions of the sequence tagger of C&C tools (Curran et al., 2007) after training it on CCGBank (Hockenmaier & Steedman, 2007). Besides, the morphological features and lemma of each token is provided as part of the lexical specification. Similar to syntactic categories, lemmas are also bootstrapped using a statistical lemmatiser (Minnen et al., 2001).

In terms of lexical-level shallow semantic annotations, GMB provides annotations of VerbNet thematic roles (Schuler, 2005), nominal animacy (Zaenen et al., 2004), named entity classes (Sekine et al., 2002), and WordNet word senses (Fellbaum, 2010). Thematic role annotations are deterministically obtained by clustering the syntactic categories of all verbs that occur in GMB corpora in terms of their arity, and mapping each such cluster to the corresponding VerbNet role label. Animacy labels are bootstrapped using a logistic regression classifier (Bjerva, 2014). Finally, named entity labels are inferred using C&C tools.

The lexical syntactic and semantic annotations of GMB are used to construct derivations at the clausal, sentence and document level. The syntactic parser of C&C tools, again trained on CCGbank, is used to obtain sentence-level syntactic derivations. GMB derivations are parsed with a CCG that assumes the schematic conjunction category, function application, first and second order harmonic and cross composition, type-raising combinators, and certain type-changing operations. All slash modalities are also implemented.

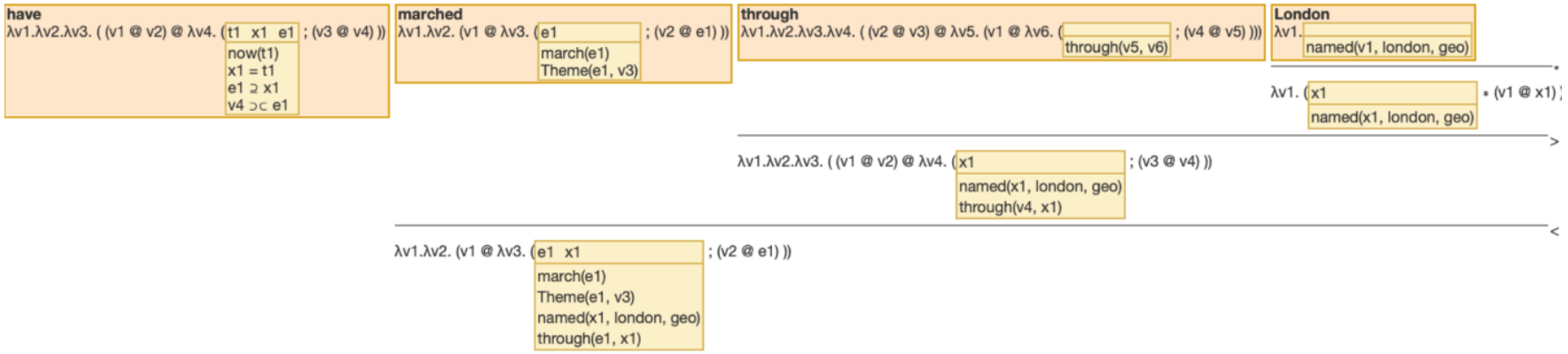


Figure 4.15: GMB dataset explorer visualisation of the S-DRT annotation of the clause '*have marched through London*' from the first sentence of GMB p.00-d.0018.

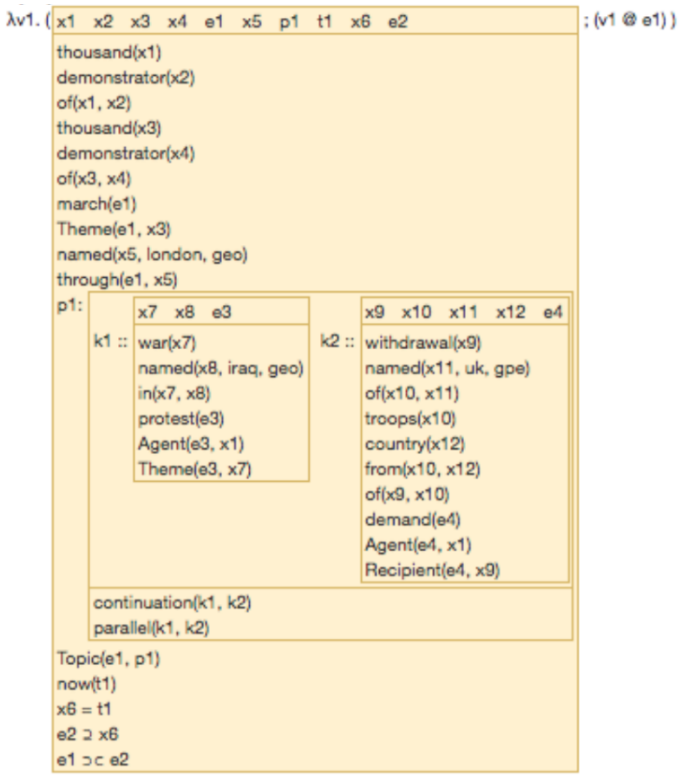


Figure 4.16: GMB dataset explorer visualisation of the S-DRT annotation of the sentence '*Thousands of demonstrators have marched through London to protest the war in Iraq and demand the withdrawal of British troops from that country.*' from GMB p.00-d.0018.

The S-DRS meaning annotations are then bootstrapped on provided syntactic derivations using a rule-based DRS parser, Boxer, by projecting DRS meaning representations to each level of derivation to obtain annotations of compositional meaning. Figure 4.15 illustrates how GMB represents compositionality at the clausal-level. Sentence-level meaning annotations, an example of which is shown in Figure 4.16, follow the method of continuation from *Section 4.6, p.46*. This enables document-level S-DRT representations to be constructed through function application of successive sentence-level meaning annotations for each GMB document.

The PMB annotation procedure diverges from GMB due to its parallel nature. In PMB, the overall annotation methodology is to bootstrap syntactic and semantic annotations on English sentences, then obtain word alignments between English sentences and their German, Italian, and Dutch translations, and eventually project lexical-level English annotations to translated sentences. Distinctive from GMB, PMB uses EasyCCG (Lewis & Steedman, 2014a) to bootstrap syntactic derivations. The lexical semantics annotations are labelled using a universal label set (Abzianidze et al., 2017). Sentence-level S-DRT annotations are obtained using Boxer as in GMB.

In terms of inter-language annotation projection, syntactic categories are first trans-

ferred to their aligned counterparts in target languages at the lexical-level. Then, a brute-force search is applied by modifying assigned lexical syntactic categories with an enumeration of variants that incorporate all possible slash directions, which yields a set of syntactic category candidates, to account for variation in word order among languages. For each token, only the syntactic category that result in a derivation that yield the source English sentence-level meaning representation in the target language are kept, while pruning other category candidates. Similar to GMB, the resulting bootstrapped annotations are selectively corrected by human experts.

CHAPTER 5

SEMANTIC ANALYSIS

This chapter reviews the semantic parsing models from the literature that adopt CCG as the grammar formalism. The problem that they tackle is to build an interpretation, either compositionally or as part of a sequence-to-sequence formalisation of the task, for given natural language text. The output interpretations which such models predict are either executable machine-interpretable FOPL logical forms or expressive meaning representations such as DRS. The input text also exhibits structural variation given the use case the model is developed for. Input is either information-seeking queries, open-domain sentences or full documents that are composed of individual sentences.

The models that are reviewed here are different from those that is presented in *Chapter 3 Syntactic Analysis* in the sense that syntactic disambiguation is only a part of the problem, which is essential to obtain an adequate interpretation for the given input since disambiguated syntactic analysis guides semantic compositionality through the syntax-semantics interface. To implement a semantic parser that constructs interpretation compositionally, one would need a grammar that accepts strings of the modelled language and the domain, but this time the lexicon should also specify the meaning for lexical constituents.

As it is reiterated in the previous chapters recurrently, lexical items of the CCG lexicon encode both the syntactic sub-categorisation of lexical constituents and their corresponding meaning. Although, crafting such elaborate lexicons is a tedious task, which is sometimes impossible to do manually considering the open-domain nature of the input, or at least challenging to semi-automatically infer in cases where we lack the baseline data that labels a corpus of input utterances that is representative of the domain we are modelling. The lack of a lexicon becomes more pronounced if the semantic parser that we are building is part of a larger modularised system, such as a question answering system. In such systems, a semantic parser acts as the first step of analysis to construct a logical form that represents the meaning of the input, which is then grounded to a knowledge base or a spatio-temporal representation of the environment that an agent acts upon.

Considering these challenges that are posed in this line of research, this chapter is structured into three parts. First, *Section 5.1* presents an algorithm that generates a CCG lexicon with spurious lexical items that encode lexical meaning with the help of supervision from a small set of utterances that are annotated with gold standard logical forms. This algorithm is named GENLEX, and it over-specifies the language we are parsing, but redundant lexical items are pruned during parameter optimisation of a parameterised model. *Section 5.2* reviews the literature that uses the GENLEX algorithm

or a variation of it to design domain-specific parameterised parsers, mostly for question answering. Finally, *Section 5.3* is about semantic parsing of open-domain natural language input to DRS logical forms through rule-based interpretation projection over a parse tree, or sequence-to-sequence modelling.

5.1 Bootstrapping CCG Lexicon

GENLEX is a lexicon bootstrapping algorithm which is devised to generate lexical items of a CCG lexicon that can parse the input when we lack a grammar. It is devised to induce a CCG grammar as a joint process with parser parameter optimisation (Zettlemoyer, 2009; Zettlemoyer & Collins, 2005, 2007). The algorithm takes a natural language utterance and its interpretation as input and generates a set of lexical items by referring to a set of rule definitions. The rule definitions are essentially lexical item templates that are instantiated into fully realised lexical items given an ordered sequence of constant values. In what follows, GENLEX is illustrated with a running example.

Consider the sentence '*Alice ran the marathon*' from *Section 2.4, p.10*. Given a lexicon, Λ, which is composed of the set of lexical items, I, that are in (5.1), we can obtain the parse, π, in (5.2) if we assume a combinatory rule set which is composed of forward and backward application. The problem we are tackling is the case where Λ is not available but we have access to an annotation that is a 2-tuple, where the first item is the utterance, υ, and the second item is the logical form, LF, that represents the meaning of υ. An example of such an annotation is shown in (5.3). We would like to induce a lexicon, I', which is composed of a set of lexical items, I', where $I \subseteq I'$.

(5.1) alice := NP : *alice′*
ran := (S\NP)/NP : $\lambda x.\lambda y.$*run′*xy
the := NP/N : $\lambda x.x$
marathon := N : *marathon′*

(5.2)

Alice	ran	the	marathon
NP	(S\NP)/NP	NP/N	N
: *alice′*	: $\lambda x.\lambda y.$*run′*xy	: $\lambda x.x$	: *marathon′*
			NP (>)
			: *marathon′*
		S\NP (>)	
		: $\lambda y.$*run′marathon′*y	
	S (<)		
	: *run′marathon′alice′*		

(5.3) $\langle \upsilon$: *Alice ran the marathon*, LF : *run′marathon′alice′*$\rangle$

The underlying principle of GENLEX is to describe the lexicon, Λ, using a set of templatised lexical items. We can define a templatic CCG lexicon, Λ^T, as a set of lexical item templates, I^T. For instance, the items in I above are templatised as in (5.4), which makes up I^T to define a generalised form of Λ. Here, surface form specifications

(e.g. '*alice*' in the first item) and logical form constants in semantic type specifications (e.g. *run′* in the assigned semantic type of the second item $\lambda x.\lambda y.$*run′*xy) in lexical items of I are templatised using variables, c_n, where n is the index of the variable.

(5.4) $c_1 := \mathsf{NP}: c_2'$
$c_1 := (\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}: \lambda x.\lambda y.c_2xy$
$c_1 := \mathsf{NP}/\mathsf{N}: \lambda x.x$
$c_1 := \mathsf{N}: c_2'$

Lexical item templates are expressed as functions. Consider the set of functions that are in (5.5), where the arity of each function, n, defines the number of constant values that should be provided to the function as input for it to yield a fully realised lexical item. For instance, the first template expects two arguments to be fully realised. As shown in (5.6), if this function takes the ordered set of constant values, C, where $C = \{alice, alice\}$, as input, the original lexical item that this template is generated from is recovered through successive function application steps after applying β-reduction and α-conversion transforms.

(5.5) $\lambda c_1.\lambda c_2.(c_1 := \mathsf{NP}: c_2')$
$\lambda c_1.\lambda c_2.(c_1 := (\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}: \lambda x.\lambda y.c_2xy)$
$\lambda c_1.(c_1 := \mathsf{NP}/\mathsf{N}: \lambda x.x)$
$\lambda c_1.\lambda c_2.(c_1 := \mathsf{N}: c_2')$

(5.6) $\lambda c_1.\lambda c_2.(c_1 := \mathsf{NP}: c_2')(alice\, alice)$
$\lambda c_1.(alice := \mathsf{NP}: c_1')(alice)$
$alice := \mathsf{NP}:$ *alice′*

We can sub-categorise the constant values in C. Given the structure of a lexical item, in their templatic forms, variables can either hold the value of a surface form specification or a logical form constant. The first set, W, is composed of constant values that realise surface form specifications, which are now represented with indexed variables, w_n. Similarly, the elements of the latter set of logical form constants, L, realise the variables that are represented as l_n. This sub-categorisation of constants are reflected in the variables of the templatic lexical items in (5.7).

(5.7) $\lambda w_1.\lambda l_1.(w_1 := \mathsf{NP}: l_1')$
$\lambda w_1.\lambda l_1.(w_1 := (\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}: \lambda x.\lambda y.l_1xy)$
$\lambda w_1.(w_1 := \mathsf{NP}/\mathsf{N}: \lambda x.x)$
$\lambda w_1.\lambda l_1.(w_1 := \mathsf{N}: l_1')$

In this context, a CCG lexicon is bootstrapped using the function GENLEX(υ, LF, I^T), which takes an utterance, the logical form annotation for that utterance, and a set of functions that generate lexical items given an ordered set of constants as input and yields the set of fully realised lexical items.

In the original form of GENLEX, as presented by Zettlemoyer and Collins (2005), the set of surface form constants, W, is obtained by computing all possible sub-strings in υ. For instance, for the given sentence '*Alice ran the marathon*', W is

computed as {'*alice*', '*alice ran*', ..., '*ran the marathon*', '*the marathon*', ...}. The set of logical form constants, L, is composed of all possible subsets of the set of logical form constants that appear in LF. For example, the annotated logical form *run′*(*marathon′alice′*) yields the logical form constants $\{\{\textit{alice}'\}, \{\textit{marathon}'\}, \ldots, \{\textit{run}', \textit{marathon}'\}, \ldots\}$.

Then, each permutation of constant values in W and L is applied to the functions in I^T. The resulting set of fully realised lexical items yields the bootstrapped grammar $'$. To illustrate, if we assume the set of functions from (5.7) are the elements of I^T, then $'$ contains the lexical items in (5.8) for the utterance annotation in (5.3).

(5.8) alice := NP : *alice′*
alice := NP : *marathon′*
ran := (S\NP)/NP : $\lambda x.\lambda y.$*run′*xy
ran the := (S\NP)/NP : $\lambda x.\lambda y.$*marathon′*xy
the := NP/N : $\lambda x.x$
the marathon := NP/N : $\lambda x.x$
marathon := N : *marathon′*
marathon := N : *alice′*

Notice that the bootstrapped lexicon, $'$, over-specifies the language that is defined by the original grammar, Λ. Half of the lexical items in (5.8) that are generated by GENLEX are spurious (e.g. alice := NP : *marathon′* does not appear in any valid derivation for the utterance). The strategy that Zettlemoyer and Collins (2005, 2007) employ is to prune the set of bootstrapped lexical items by eliminating those items that do not yield a derivation to an interpretation that is equivalent to the annotated logical form, LF, for the utterance, v, which is input to GENLEX. This strategy keeps a minimal set of the lexical items in $'$ and ensures that those items define a grammar that can parse the input utterance into target logical form while pruning redundant items.

The literature that is reviewed in *Section 5.2, p.58* either employs this lexicon induction algorithm as demonstrated here as part of their model training algorithm, or they implement an approach that is derived from this technique.

5.2 Mapping Sentences to Executable Interpretations

Zettlemoyer and Collins (2005) present the first research on a learning algorithm that is used to induce CCG semantic parsers by jointly bootstrapping a lexicon and the parameters of the parsing model. The algorithm implements a supervised learning method. The supervision is provided over a minimal set of annotated examples that are pairs of utterances and their gold-standard logical forms, similar to those that are shown in *Section 5.1, p.56*. The output of the algorithm is a parameterised CCG lexicon.

The parameterisation of the lexicon is attained by extending the definition of the base CCG lexicon, which was introduced in *Section 2.1, p.5* and has been used so far in

the previous chapters. A **Probabilistic CCG** (PCCG) is defined as a lexicon, Λ, that is coupled with a parameter vector, θ. The parameter vector holds values for each individual feature, f. The number of features, d, that the model uses determines the dimension of the parameter vector, where $|\theta| = d$. At training time, the aim is to optimise these parameters to obtain a model that is representative of the distribution of the syntactic structures that are seen in the data that provides the supervision. Similarly, at inference time, the parameters are used to disambiguate derivations that yield ambiguous interpretations by finding the most probable derivation among a set of derivations that arise due to lexical and spurious ambiguity. Zettlemoyer and Collins (2005) formally define the conditional distributions that are computed using PCCGs as follows:

(5.9) $P(LF, \pi|S)$

Above, LF is a logical form that the derivation, π, yields for the given natural language sentence, S. Within this context, a derivation is defined as a series of lexical scan and combinatory rule application steps. For instance, (5.2) presents a derivation, π, that yields the logical form, LF, which is *run′*(*marathon′alice′*), through 7 steps (4 lexical match, 3 combinatory rule application), for the given sentence, S, '*Alice ran the marathon*'. Given the predictive power of PCCG as such conditional distributions, the statistical approximation of a semantic parse with PCCG is defined as follows, as an extension of the syntactic parsing models that are presented in *Section 3.3, p.23*:

(5.10) $$P(LF, \pi|S; \theta) = \frac{e^{\bar{f}(LF,\pi,S)\cdot\bar{\theta}}}{\sum\limits_{(LF,\pi)} e^{\bar{f}(LF,\pi,S)\cdot\bar{\theta}}}$$

Similar to the syntactic parsing models, features are lexical or contextual predicates. Above, $\bar{f}$ is a function that takes in the logical form, the derivation, and the sentence and counts certain sub-structures that the feature f is defined on in these inputs to find the feature value. Zettlemoyer and Collins (2005) only utilise lexical features. That is, each feature, f, is the count of the occurrence of lexical items in a given derivation. Also, the dimension of the parameter vector, d, is the size of the lexicon, $|\Lambda|$. Using such features, the most probable interpretation, LF, given a sentence S and parameter vector, θ, is inferred with the following log-linear model:

(5.11) $$\arg\max_{LF} P(LF|S; \theta) = \arg\max_{LF} \sum_{\pi} P(LF, \pi|S; \theta)$$

Here, the syntactic derivation, π, is a hidden variable, where the summation on the right-hand side is over all derivations that yield the target interpretation, and the $\arg\max$ is over all logical forms the grammar yields for the sentence, S. At training time, the parameters in θ are estimated using the stochastic gradient ascent algorithm (LeCun et al., 1998), by differentiating the log-likelihood of each example from the training set with respect to the likelihood of the derivation that the model yields at each gradient update.

The learning algorithm that Zettlemoyer and Collins (2005) present is composed of three procedures; parsing, parameter estimation, and lexical item generation. The

lexicon is either initialised to be empty or includes certain seed lexical items (e.g. items that can parse function words). The parameter vector is instantiated with a uniform distribution. For each example in the training set, new lexical items are bootstrapped using GENLEX and added to the lexicon. Then, the most probable parse is obtained given the updated lexicon using a dynamic programming approach, such as the CYK algorithm. The lexicon is pruned at this stage by removing items that do not occur in the most probable derivation from the set of items that GENLEX produced for the sentence. After updating the lexicon using all the sentences in the training set, the parameters are re-estimated at the end of the epoch. This process is repeated for the predefined number of learning epochs.

Zettlemoyer and Collins (2007) present an online variation of this algorithm where the parameter estimation is carried out after the completion of the lexicon update with each example. Also, the online learning algorithm implements additive perceptron updates instead of stochastic gradient ascent for parameter estimation. Both studies present empirical results on lexicon bootstrapping for semantic parsing of the queries from two question answering datasets, Geo880 and Jobs640 (Tang & Mooney, 2001).

Kwiatkowksi et al. (2010) present a generalised form of this approach through higher-order unification. The above discussed GENLEX algorithm employ a set of rule definitions that are tied to the language and the domain of the data that is being parsed. Generalised higher-order unification eliminates the need to create language-specific templatic rule sets by hand. In this approach, first the lexicon is initialised with a single entry that can parse the full sentence, such as:

(5.12) alice ran the marathon := S : $run'marathon'alice'$

This single lexical item can parse the full sentence scope, but it does not generalise to other sentences. The unification technique that Kwiatkowksi et al. (2010) introduces employs operations that split such lexical items into multiples, for instance:

(5.13) alice ran := S/NP : $\lambda x.run'xalice'$
the marathon := NP : $marathon'$

In order to constrain the over-generation of spurious lexical items, certain split restriction rules are applied. These rules implements a control on split operations over the generated logical forms (e.g. vacuous variables are not permitted) and syntactic categories (e.g. type-dependent constraints on generated complex categories are adopted). Given the constrained higher-order unification rules, Kwiatkowksi et al. (2010) present an online learning algorithm similar to the one employed by Zettlemoyer and Collins (2007) to bootstrap semantic parsers without using a base templatic rule set.

The efficiency of the bootstrapped parser is correlated with the ambiguity that is encoded in the lexicon. Hence, follow-up research devises techniques to control this ambiguity by minimising the generated lexicon size. For instance, Kwiatkowski et al. (2011) conflate lexical items over templatic logical forms, which are differentiated by the set of constants that realise templates to target meanings. In this context, such templatic lexicons are called **factored lexicons** that are composed of a set of lexical

item templates and a set of lexemes. Lexical templates are duals of the templatic lexical items that are presented in *Section 5.1, p.56*. A **lexeme** is a tuple of surface form specification and a set of logical form constant that fully realise a templatic lexical item. The maximal factoring of a lexicon is attained by finding the set of unique lexemes and template pairs that can fully realise a CCG lexicon. Kwiatkowski et al. (2011) extend the online learning algorithm that is used by the previous research to learn a compact maximally factored lexicon.

In this problem space, not all research focuses on question answering. For instance Artzi and Zettlemoyer (2011) apply the methodology that stems from GENLEX to model a semantic parser for conversations. As part of the task of parsing open-domain or goal-oriented conversations and instructions that can be grounded in well-defined environments in order to navigate in physical space, open-endedness have to be captured in the meaning represented by the generated logical forms. It is observed that the lexicons that are induced for such domains yield noisy and overly ambiguous grammars. To that end, Artzi et al. (2014) present global voting and pruning schemes that dynamically adjust the lexicon size in online learning. Throughout the online learning, after each learning step, corpus-level statistics are gathered to make decisions on eliminating or populating lexicon entries. Artzi et al. (2014) show that global decision rules significantly reduce the lexicon size, while outperforming all previous mentioned lexicon size restraining techniques.

The rest of the research that is reviewed below aims to ground the logical forms that are mapped from sentences to context. Artzi and Zettlemoyer (2013) parse instructions into intermediate representations which can then be grounded to spatial real-world context. Here, the modelled language contains sentences with imperative constructions that have the semantics of requesting a certain action in an environment that contains visible real-world objects that an agent can act on. The logical form annotations that represent the semantics of instructions adopt neo-Davidsonian event semantics, and GENLEX is used to induce a lexicon with such representations during parameter optimisation. This work also models the execution of disambiguated logical form expressions against a given environment to simulate the completion of the requested action. Therefore, the model carries out joint inference on disambiguating the logical forms that the input natural language sentence can be mapped to and grounding and executing the meaning based on the states of the modelled environment and the agent.

In terms of parameterised grammar induction for question answering, the research mostly employs an approach where questions are first parsed into intermediate representations, then these representations are grounded in knowledge bases using structure and graph matching to retrieve answer candidates. The grounding algorithms exhibit variation. For instance, Kwiatkowski et al. (2013) introduce a structure-based ontological matching algorithm, where the logical constants and predicates of the disambiguated interpretation that is output by the parameterised semantic parser are successively replaced with constants from the ontology. In this work, the knowledge base that the meaning is grounded in is the Freebase (Q. Cai & Yates, 2013), where the real-word knowledge is represented as triples that encode relationships between entities. The relationship specifiers match the logical predicates in the logical forms and the entities realise the constants.

Reddy et al. (2014) are also concerned with grounding FOPL representations to

Freebase. However, they formalise the problem of grounding as graph-matching. Given the triples of Freebase, the information that is encoded in the knowledge base can be represented as a directed graph. The insight of Reddy et al. (2014) is to map the disambiguated logical forms into graph-structures as well, where the nodes of the graph are typed in order to differentiate the logical constants and predicates that require grounding in the ontology. The algorithm matches the relationships and entities from the ontology graph to the typed nodes of the logical form graph to obtain a grounded graph which can recover an answer to the question.

Lastly, Reddy et al. (2016) introduce the problem of lacking access to examples of natural language queries that are annotated with gold standard logical forms that can be used to induce parameterised lexicons. They use dependency trees as proxies to generate logical form annotations. For that, the sentences that are in the training set are first parsed into the corresponding dependency structure using a syntactic parser. Then, the dependency parse is binarised into an s-expression that represents the logical predicates and arguments in interpretation. Finally, s-expressions are mapped into λ-calculus expressions that annotate the meaning of each sentence in FOPL. This approach generalises grammar induction across domains and languages as long as there is access to an acceptably well performing dependency parser for the target language, while eliminating the need for manually annotated interpretations over a set of training examples.

5.3 Discourse Representation Structure Parsing

DRS parsing is the natural language understanding task where an input open-domain sentence or document is mapped to the DRS logical form that represents the meaning of the input. With the development and release of two large-scale meaning banks, GMB and PMB, which are reviewed in *Section 4.7, p.50*, the research on DRS parsing gained traction. Prior to the release of these datasets, the focus was on developing monolingual analysers for English that projected semantic analysis on top of syntactic derivations that are output by a syntactic parser, such as the C&C parser that is reviewed in *Section 3.3, p.23*. After obtaining the wide-coverage meaning banks, the line of research shifts to developing sequence-to-sequence neural network models while greater emphasis is given to multilingual analysis.

Two of the early attempts at developing a wide-coverage DRS parser are by Le and Zuidema (2012) and Bos (2015). Le and Zuidema (2012) present a statistical learning algorithm that approximates a lexicon given triples of a sentence, the syntactic parse of the sentence, and a semantic graph. The box-notation, which is presented in *Section 4.2, p.33*, is not an adequate representation as a data structure in implementation. Therefore, Le and Zuidema (2012) first map DRS meaning representations in box-notation to graph structures. In this context, underspecified DRS logical forms are named as partial graphs. Two partial graphs can merge using a set of deterministic rules. These merge operations are parallel to the combinatory rules that combine the semantic types of two adjacent constituents. The learning algorithm of Le and Zuidema (2012) disambiguates the most probable set of partial graphs that are assigned to the lexical constituents of the given sentence which yield the sentence-level semantic graph over the given syntactic parse. On the other hand, Bos (2015) presents a rule-based system,

named Boxer, that implements the semantics of combinatory rules. Boxer projects DRS meaning representations on the parse tree for all steps of the derivation in a top-down fashion given a syntactic parse for the input. The system is used to bootstrap GMB logical form annotations on C&C syntactic parser output.

Successive research mainly formalises the task as a translation problem and focuses on sequence-to-sequence modelling using encoder-decoder models. In such approaches, the problem is defined as translating a sequence of words that compose the given natural language text to an expression of the logical form language, which is the DRS language. In these end-to-end models, the syntactic derivation and compositionality are not explicitly modelled on the input side, whereas some models only encode certain syntactic features (e.g. POS or syntactic category of each word) of the input sentence together with the natural language text. Given the architecture of these models, the predicted DRS meaning representations are first linearised and tokenised to obtain the output representations. Their logical form linearisation methodology is usually a variation of the linearisation technique that is described in *Section 6.1, p.68*.

The first research that adopts this approach is by J. Liu et al. (2018). Their focus is on testing various hierarchical decoder architectures in an encoder-decoder model to parse sentences of GMB into sentence-level DRS logical forms. The choice of design of the experimental decoder architectures stems from the structure of the DRS representation. The baseline model that they present is a naive LSTM sequence decoder that does not differentiate the likelihood of the predicted DRS conditions, where all tokens that are step-wise predicted as part of the output sequence are treated uniformly. Two experimental models are tested against the baseline. First is a shallow structure model that implements a copy mechanism to predict DRS conditions that are local to the context of the sentence and uses an insertion mechanism to fill in non-local tokens of the output sequence, such as thematic role or discourse relation values. The second experiment model is a deep structure model that predicts a skeletal DRS first and then fills in the referents and conditions in this skeletal structure sequentially. Their findings show that hierarchical decoding improves DRS parsing accuracy compared to the baseline, whereas the deep structure model performs better than the shallow structure decoder.

J. Liu et al. (2019a) are concerned with parsing multi-sentence documents of GMB into document-level logical forms rather than sentence-level processing solely. Increased processing scope from sentence to document-level means larger DRS meaning representations to predict and recovery of a more ambiguous output sequence. They again use an encoder-decoder model for this task. This time, the input natural language documents are encoded with bi-LSTM units that can learn from both the left and right context of a given word in each time-step. The decoder is hierarchically structured as a 3-layered stack similar to their prior research. The first layer predicts the skeletal structure of meaning by outputting the S-DRS and DRS scopes and predicate-argument structure. The second layer predicts the relations that are encoded as part of discourse conditions. The final layer fills in all classes of variables in the structure that is composed by the previous layers. In addition, between the encoder and decoder stack hidden-layers multi-attention heads and copy mechanisms are employed, so that the decoder can condition on the input natural language text and also copy free-form predicates from input that are encountered as arguments of DRS conditions on the output.

Fu et al. (2020) present an extension to document parsing by harnessing the syntactic structure of the input and the skeletal graph-structure of the output using Graph Attention Networks (GAT; Veličković et al., 2018). Previous DRS parsers did not implement any special treatment of the natural language input, only focusing on the hierarchical step-wise prediction of the output structure. Fu et al. (2020) show it is possible to represent both the input documents and their corresponding DRS meaning as a tree-structure, which is an acyclic undirected graph. GATs are known to provide improved encoding for such graph structures. In this case, the input natural language text is first mapped to a dependency graph by parsing it with a dependency parser. The output DRS is parsed into a tree-structured representation, similar to the tree-based DRS representation from *Section 6.2, p.70*. When the encoder and the decoder stacks implement GAT networks that encode tree-structured input and output improved document-level parsing accuracy is obtained compared to the results of J. Liu et al. (2019a).

The rest of the research that is reviewed below uses PMB, hence they report results on multilingual sentence-level DRS parsing. In particular, the First Shared Task on Discourse Representation Structure Parsing (Abzianidze et al., 2019) yielded a variety of models that investigate the effect of input encoding strategies on DRS parsing.

van Noord, Abzianidze, Toral, et al. (2018) is the first research that tests the impact of variation in encoding of the natural language input in a bi-LSTM encoder-decoder model with general attention implemented between the encoder and decoder stacks. The test cases implement variation in the encoder stack by modelling character-level, word-level, and frequency-based representations. In the character-based model, input sentences are modelled over sequences of individual characters. For the word-based model, space-tokenised sentences are encoded using pre-trained GloVe embeddings (Pennington et al., 2014). For the frequency-based approach, Byte-Pair Encoding (BPE; Sennrich et al., 2016) is used to find a sub-word unit tokenisation of the words of the input using a pre-trained co-occurence based tokenisation model. Empirical results by van Noord, Abzianidze, Toral, et al. (2018) demonstrate that BPE representation does not yield any improvements in DRS parsing over the baseline, while character-level models outperform word-based representations. One interesting finding of the experiments is joint encoding of both character and word-level representations by concatenating both vector representations to encode the input results in significant parsing accuracy improvements.

All the above mentioned models use bi-LSTM architectures as the parsing model. J. Liu et al. (2019b) is the first research that tests using a Transformer model (Vaswani et al., 2017) in DRS parsing. The natural language input is encoded as a lowercase-folded sequence of tokens that Moses tokeniser (Koehn et al., 2007) yields. The output sequences encode DRS logical forms in linearised clausal form representation, as presented in *Section 6.1, p.68*. The experiment results show that the Transformer model that encodes input with word-level representations can perform similar to LSTMs with character-level encoders on predicting certain sub-structures of DRS. Other research that implements a novel model architecture for DRS parsing is by Evang (2019). They present a stack-LSTM shift-reduce parser where the transition decisions are conditioned on the vector representation of parser states.

van Noord et al. (2019) is the first work that encodes lexical, syntactic (e.g. PoS

and categories) and semantic annotations together with sentences in a multi-encoder architecture. They use a bi-LSTM parsing model, where both the input and output are represented as sequences of characters. The experiments aim to exploit the encoding of syntactic and semantic features that are obtained for the natural language input using standalone analysers. For instance, POS tags and morphological features are extracted using the Stanford NLP toolkit (Manning et al., 2014), and the syntactic categories are gathered by parsing the sentences with the easyCCG parser (Lewis & Steedman, 2014a). In terms of model architecture, two experiment cases are tested. First, features for each word of the sentence is concatenated together with the natural language sentence. Second, feature layers are encoded separately from the input sentence. The results of the experiment show that multi-encoder architectures, where linguistic features are jointly encoded with input text, improve parsing performance compared to standalone encoding of the surface form of the input.

Finally, the findings of van Noord et al. (2020) are particularly relevant to the discussion of the experiments that are presented in *Chapter 7 Experiments*. They compare parsing performance when using standalone pre-trained input representations (Devlin et al., 2019; Y. Liu et al., 2019; Peters et al., 2018) together with character-level and word-embedding representations (Grave et al., 2018; Pennington et al., 2014) in single and dual encoder setups. The evidence from this work suggests that representations from pre-trained models outperform standalone character-level and word-embeddings input representations while parsing with bi-LSTM architectures. Joint encoding of character-level representations and semantic features yields additional improvements. Among those pre-trained models, $\text{BERT}_{\text{base}}$ (Devlin et al., 2019) delivers the best performing standalone input representations for both bi-LSTM and Transformer parsers.

CHAPTER 6

CROSS-LEVEL TYPING THE LOGICAL FORM

This chapter is on assigning types to the logical form to obtain typed representations that can be used in devising novel computational modelling approaches for semantic parsing of natural language text to meaning representations. The approach is based on first providing a formal definition of the syntax of the logical form language in hand and then inducing a type system that is solely dependent on the syntactic description of the defined language. In that sense, a generalised form of the typing methodology that is presented in this chapter is also applicable to FOPL meaning representations that are presented in previous chapters.

The focus in this chapter is on typing DRS meaning representations and their segmented extensions, especially as they are annotated in GMB. Segmented-DRS representations jointly encode linguistic phenomena that are traditionally studied at different levels (e.g. tense at sub-lexical level, words sense or named entity classes at lexicon, predicate-argument structure at sentence-level, and rhetorical relations at discourse-level). By typing the DRS meaning representations, it is possible to represent annotations for all these phenomena as functionally equivalent tokens of a DRS expression with respect to the syntax of the DRS language. Hence, the introduced logical form typing is cross-level as it can holistically capture linguistic phenomena from lexicon up to discourse.

In *Section 6.1*, the chapter starts by presenting two notations, linearised and clausal form, which are used in representing DRS logical forms that are alternatives to the box-notation that is introduced in *Chapter 4 Representing Open-Domain Meaning*. These representations are used all through the rest of this chapter to illustrate the method of obtaining typed sequence representation in downstream parsing tasks. *Section 6.2* defines a context-free grammar that recognises DRS meaning representations that are used in annotating GMB. This grammar is then used in *Section 6.3* to introduce the type assignment procedure that is based on the syntax of the defined logical form language.

The rest of the chapter presents the scope of a selection of problems from the semantic parsing domain in which type assigned logical forms provide efficient modelling solutions. *Section 6.4* provides the definitions for full and partial templatisation of the logical form by benefiting from the logical form type assignment, which is illustrated to be closely related to the Masked Language Modelling (MLM) pre-training objectives. Finally, *Section 6.5* is on the re-formalisation of supertagging as a tagging task to jointly disambiguate syntax and semantics during the lexical scan phase of parsing to attain a genuinely compositional derivation of the interpretation.

6.1 Linearised and Clausal Form Notations

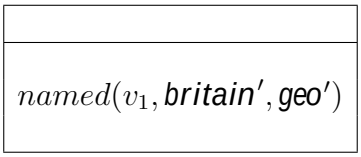
Section 4.2, p.33 introduced the box-notation to represent DRSs. So far, box-notation has been adopted as the sole representation to work with DRSs in this thesis. However, alternative notations exist. Other than the box-notation, two common DRS representations are the linearised and clausal form. Below, linearised and clausal form representations are informally defined over running examples. These alternative representations will help to introduce the typed DRS logical form in *Section 6.3* across various levels of analyses and also to evaluate the models that are presented in *Chapter 7 Experiments*.

The first one, **linearised form**, is useful in obtaining tokenised sequential representations for a given DRS. Such flat representations are utilised in encoding input and output sequence representations of sequence-to-sequence models, such as LSTM and Transformer-based encoder-decoder, or decoder-only model architectures. Most of the models that are reviewed in *Section 5.3, p.62* use linearised sequence representations.

Given a DRS in box-notation, the corresponding linearised form is obtained as a string. For instance, (6.1) is the box-notation representation for the lexical meaning of the named entity '*Britain*'.[1] The λ-abstracted DRS is expressed as a flat string, as in (6.2), after assigning unique string functor names to the sub-structures (e.g. DRS or S-DRS), the logical predicates (e.g. *named*) and the operators (e.g. negation or implication) that are observed in the meaning representation.

GMB and PMB annotations specify particular functor names and have a unique representation of scoped structures. (6.3) shows the annotated lexical semantics in GMB linearised form for the same named entity from above. Note that in the linearised notation that GMB adopts, DRS and S-DRS condition and referent sets are represented in bracketed scope, whereas discourse partitioning is represented as a literal. In order to compose sequence representations in modelling, the functionally meaningful tokens of linearised form are tokenised, as in (6.4), while discarding commas or other tokens that do not contribute to the overall semantics of the lambda expression.

(6.1) $\lambda v_1.$



(6.2) $\lambda v_1.(drs(conditions() \wedge referents(named(v_1, \textsf{britain}', \textsf{geo}'))))$

(6.3) $lam(v_1, drs([], [named(v_1, \textsf{britain}', \textsf{geo}')]))$

(6.4) $lam(,\, v_1,\, drs(,\, [,\,],\, [,\, named(,\, v_1,\, \textsf{britain}',\, \textsf{geo}',\,),\,],\,),\,)$

The latter, **clausal form**, is used in both encoding the output meaning representation in semantic parsing and also in evaluating and benchmarking model performance using

[1] The lexical logical form annotation is for the occurrence of the word '*Britain*' in the sentence '*The protest comes on the eve of the annual conference of Britain's ruling Labour Party in the southern English seaside resort of Brighton*' from GMB p.00-d.0018.

automated evaluation methods that are conventionalised in the literature. Likewise, all the sequence-to-sequence models from *Section 5.3, p.62* transform their model predictions to clausal form to provide cross-comparable parsing performance metrics.

In clausal form, a given DRS in box-notation is converted to a set of clauses using the transform algorithm that is presented by van Noord, Abzianidze, Haagsma, et al. (2018). For example, (6.5) represents the meaning of the sentence '*They marched from the Houses of Parliament to a rally in Hyde Park*' in box-notation.[2] The corresponding clausal form notation representation for this sentence is in (6.6).

A clause is a tuple. Each referent of a DRS is mapped to a 3-tuple where the first item specifies the label of the DRS that the referent is a member of (e.g. b_1 that maps to a DRS that is labelled with π_1), the second item is a predicate that denotes the variable is a referent (e.g. REF), and the last item is the variable that corresponds to the discourse referent.[3] The conditions of a DRS are represented by a 4-tuple. The first item of the tuple, like in the case of referents, always specifies the membership relationship of conditions to corresponding DRSs. The remaining items specify the logical predicate and arguments of the discourse condition (e.g. the clause 'b_1 Theme e_1 x_1' corresponds to the neo-Davidsonian encoding of the thematic role of the verb over a reified event variable in predicate-argument structure, such as $Theme(e_1, x_1)$).

(6.5) $\lambda v_1.(\pi_1 :$

$x_1, x_2, x_3, x_4, x_5, x_6, e_1, t_1, t_2$
$pred(x_1, thing, n.12)$
$pred(x_2, house, n.01)$
$pred(x_3, parliament, n.01)$
$pred(x_4, rally, n.01)$
$pred(e_1, march, v.01)$
$named(x_5, park, geo)$
$named(x_6, hyde, geo)$
$role(e_1, x_1, theme)$
$rel(x_2, x_3, of)$
$rel(x_5, x_6, eq)$
$rel(e_1, x_2, from)$
$rel(e_1, x_4, to)$
$rel(e_1, x_5, in)$
$rel(e_1, t_2, \subseteq)$
$rel(t_2, t_1, \prec)$
$t_1 = now$

$\oplus (v_1 @ e_1))$

[2] Sentence-level logical form annotation is from GMB p.00-d.0018.

[3] In clausal form, S-DRSs are also captured with labels k_n and the DRSs that are in the scope of the S-DRS are represented with clauses that are in the form "k_0 DRS b_1", which reads as b_1 is a DRS that is in the scope of the S-DRS that has the label k_0.

(6.6)

$b1$ REF x_1	$b1$ thing n.12 x_1	$b1$ Eq x_5 x_6
$b1$ REF x_2	$b1$ house n.01 x_2	$b1$ From e_1 x_2
$b1$ REF x_3	$b1$ parliament n.01 x_3	$b1$ To e_1 x_4
$b1$ REF x_4	$b1$ rally n.01 x_4	$b1$ In e_1 x_5
$b1$ REF x_5	$b1$ Geo park x_5	$b1$ time t_1 now
$b1$ REF x_6	$b1$ Geo hyde x_6	$b1$ TIN e_1 t_2
$b1$ REF e_1	$b1$ march v.01 e_1	$b1$ TPR t_2 t_1
$b1$ REF t_1	$b1$ Theme e_1 x_1	
$b1$ REF t_2	$b1$ Of x_2 x_3	

6.2 The Logical Form Language

The logical form type assignment methodology that is presented in 6.3 derives a type system from a structural description of the meaning representation. DRS meaning representations are well-formed expressions of a logical form language. In order to craft a type hierarchy, the grammar of the logical form language that accepts DRS representations in linearised form should be defined. In this section, a language that is used in annotating DRS logical forms in GMB is introduced.

A CFG, named γ for convenience, that describes the syntax of the logical form language and accepts the strings of linearised and tokenised DRS annotations, such as (6.4), is in Figure 6.1. This grammar accepts the annotation strings for all levels of analysis: lexical, sentence, and discourse-level. The lexical logical form annotations of GMB are λ-calculus expressions that are abstracted over DRS expressions. Likewise, sentence-level annotations are also λ-abstractions since GMB adopts the method of continuation, which is defined in *Section 4.6, p.46*. On the contrary, document-level logical forms are fully specified DRS or S-DRS expressions due to the assumption that a document is a self-contained text that is composed of sentences that compositionally build a coherent interpretation that represents the discourse structure. Therefore, the top-most non-terminal of γ, which is <lf>, rewrites all the sub-structures that are observed in DRS annotations from all levels. These sub-structures are λ-terms, literals, DRS, S-DRS, and the discourse partitioning (vertical or horizontal) that is introduced by the discourse relations.

Terms, captured with the <term> rule, rewrite λ-abstracted variables and the body of the term, which is a logical form as well. Literals are rewritten with the <literal> rule and capture the discourse conditions and discourse relations. They are composed of logical predicates and a series of arguments. DRSs, captured with the <drs> rule, rewrite an arbitrary number of discourse referents and conditions, which are respectively rewritten into variables and literals by the <referent> and <condition> rules. S-DRSs are captured with the <sdrs> rule that rewrites an arbitrary number of logical forms (DRSs that are within the scope of the S-DRS) and literals that specify the discourse relations between the S-DRS scoped DRSs. The scope of the λ-terms, discourse partitioning and (S-)DRSs are marked with special terminal symbols (*lam, lab, sdrs, drs*). All these rewrite rules define the recursive structure of the DRS language.

Among the rest of the rewrite rules, only the rule that has <constant> non-terminal on the left-hand expands to an open set of terminal symbols. <predicate> rule rewrites a

<lf>	→	<term> | <sdrs> | <drs> | <partition>
		| <literal> | <variable> | <constant>
<term>	→	lam <variable> <lf>
<sdrs>	→	$sdrs$ <lf>* <literal>*
<drs>	→	drs <referent>* <condition>*
<partition>	→	lab <variable> <lf>
<literal>	→	<predicate> <lf>*
<referent>	→	<variable>
<condition>	→	<literal>
<predicate>	→	imp | $role$ | $named$ | ...
<variable>	→	v_i | x_i | e_i | t_i | p_i | t_i | π_i
<constant>	→	$annual$ | of | $continuation$ | geo
		| $theme$ | *2014* | ...

Figure 6.1: Context-free grammar that accepts the DRS annotations of GMB version 2.2.0. Non-terminal symbols are displayed in angle quotation marks, and the terminal symbols are in italic.

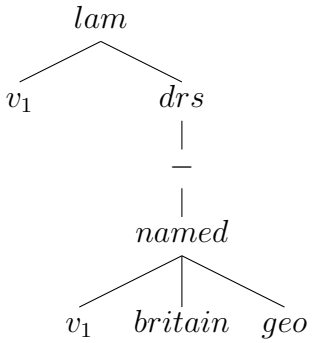
closed set of symbols which capture free-form predicates and their associated word sense (*pred*), spatio-temporal relations (*rel*), quantifiers (*not, imp, or*), modality (*nec, pos*), duplex constructions (*dup*), propositions (*prop*), discourse partitioning (*sub*), named entities (*named*), thematic role (*role*), date/time expressions (*card, timex, date*), and also the predicates of λ-calculus (*alfa, merge, app*).

The variable arguments of literals and λ-terms, and the discourse referents of DRSs are rewritten by the <variable> rule. As shown in *Section 4.6, p.46*, variables are classified as lambda, entity, event, temporal, proposition, topicality, and discourse variables. Therefore, each variable class is marked with distinct terminal symbols (e.g. $\{v, x, e, ...\}$). In theory, a given DRS expression might contain infinitely many variables. One strategy to capture the open set of variables is to define the variable indices, i, to take their value from the set of positive integers, $\mathbb{Z}+$. Alternatively, given the limited discourse structure that the sentences GMB annotate, the set of variables is defined as a closed set with a finite number of indexed variable members. In this case, all possible indexed variables that are part of this closed set is included in γ as a terminal symbol that is rewritten by the non-terminal <variable>. Henceforth, the latter strategy is adopted and variables are assumed to come from a finite set of terminal symbols.

Given this grammar, γ, a stack-based shift-reduce parser that recognises the expressions of the DRS language defined by γ can parse DRS expressions that are in tokenised and linearised form into a parse tree after postfix ordering the tokens of the expression.

Consider the tokenised lexical semantics annotation in linearised form from (6.4). The corresponding parse tree that such a parser yields for this expression is as follows:

(6.7)



Note that the grammar, γ, does not capture the parentheses and bracketing that DRS expressions, such as in (6.3), contain to mark scope. For brevity in this illustration, scope markers are intentionally omitted. They can be added in γ by including an <operator> rule that rewrites these symbols where necessary. This simplification is also carried out in the rest of this chapter.

6.3 Type Assignment on Logical Form Across Levels

This section presents an algorithm that assigns types to the logical forms. The algorithm takes a logical form, LF, as input and outputs a typed logical form, LF'. It is assumed that the LF is a meaning representation, and in its tokenised and linearised form it is a grammatical expression of the language that is defined by the grammar, γ, that is presented in *Section 6.2, p.70*.

In this context, LF is a sentence that is composed of a sequence of words such as $\{lam, v_1, drs, named, v_1, \mathit{britain}', \mathit{geo}'\}$, which is the expression that corresponds to the lexical logical form from (6.1). If Σ_γ is the set of terminal symbols of the grammar, γ, then any word, w, that appears in such sentences is a member of the set of terminal symbols, $w \in \Sigma_\gamma$. For any given $LF = \{w_1, \ldots, w_n\}$, where n is the number of words that are in the LF expression, we would like to obtain a typed logical form expression, $\mathit{LF}' = \{w_1^t, \ldots, w_n^t\}$, where t is a type that is assigned to the words of LF. Every type, t, is a member of the set of types, T. It is assumed that a type specifies a set of values that are functionally equivalent within the given syntactic description of the logical form language.

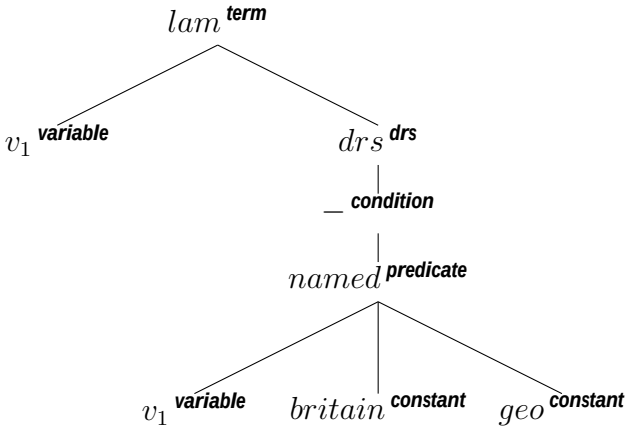
The type system that is presented here is solely derived from the syntax of the LF language as dictated by γ. In this sense, it deviates from the semantic type systems, such as the one that is used by Bos et al. (2017) to define the syntax of DRSs. Let V_γ be the set of non-terminal symbols of the grammar γ, then $\forall t \in T$ is also a member of the set of non-terminal symbols, where $t \in V_\gamma$. That is, the type inventory, T, is a closed set with 11 members, $T = \{lf, term, sdrs, drs, partition, literal, referent, condition, predicate, variable, constant\}$.

The type assignment algorithm is essentially a deterministic bottom-up parsing algo-

rithm based on the shift-reduce parser which is introduced in *Section 6.2, p.70*. The only extension to the parser is that while parsing linearised DRS expressions a type is assigned to each node of the parse tree. Given a parse tree, π, for ***LF***, every tree node, $n \in nodes(\pi)$, is assigned a type, $t \in T$, to obtain a typed node, n_t, where t is the corresponding non-terminal that rewrites the sub-tree that is covered by n.

Now, type assignment is illustrated across different levels of analysis, starting from the lexical logical form, and progressing to the sentence and discourse levels. At lexical-level, above exemplified LF expression, $\{lam, v_1, \ldots, \textit{britain}', \textit{geo}'\}$, is parsed into (6.7) with the aforementioned shit-reduce parser. With the described extension to this parser, the parse tree in (6.8) with typed nodes is obtained. Here, the non-terminals that rewrite each sub-tree are displayed as superscripts.

(6.8)



Note that the parse tree, box-notation, and linearised form representations are equivalent and can be translated into one another. For instance, the above parse tree with typed nodes yields the typed linearised representation that is in (6.9) after depth-first traversal of the tree nodes. Similarly, the corresponding typed λ-term which abstracts a DRS that can be obtained from equivalent representations is shown in (6.10).

(6.9) $lam^{\textbf{term}}\ v_1^{\textbf{variable}}\ drs^{\textbf{drs}}\ named^{\textbf{predicate}}\ v_1^{\textbf{variable}}\ britain^{\textbf{constant}}\ geo^{\textbf{constant}}$

(6.10) $(\lambda v_1^{\textbf{variable}}.$

$$\begin{array}{|c|}\hline \\ named^{\textbf{predicate}}(v_1^{\textbf{variable}}, britain^{\textbf{constant}}, geo^{\textbf{constant}})^{\textbf{literal}} \\ \\ \hline \end{array}^{\textbf{drs}}$$

$)^{\textbf{term}}$

Sentence-level meaning representations are λ-terms as well. For example, (6.11) is the typed variant of the sentence-level logical form (6.5). For brevity, hereafter, the following short-hands are used for each type: ***t*** $(term)$, ***s*** $(sdrs)$, ***d*** (drs), ***l*** $(literal)$, ***p*** $(predicate)$, ***v*** $(variable)$, ***c*** $(constant)$. Again, for simplicity, the lf, $referent$ and $condition$ types are omitted in box-notation representation. This is due to the recursive nature of the grammar, γ, where the corresponding rules for these types do not rewrite an immediate terminal symbol on their right-hand side, therefore creating nodes in the parse tree that do not map to any words of the parsed logical form expression. In a similar fashion, $partition$ type is already encoded in the structure of the S-DRS in box-notation, therefore they are not marked in the examples.

(6.11) $(\lambda v_1^{\mathbf{v}}.($

$$\begin{array}{|l|}\hline x_1^{\mathbf{v}}, x_2^{\mathbf{v}}, x_3^{\mathbf{v}}, x_4^{\mathbf{v}}, x_5^{\mathbf{v}}, x_6^{\mathbf{v}}, e_1^{\mathbf{v}}, t_1^{\mathbf{v}}, t_2^{\mathbf{v}} \quad \mathbf{d} \\ \hline \\ pred^{\mathbf{p}}(x_1^{\mathbf{v}}, thing^{\mathbf{c}}, n.12^{\mathbf{c}})^{\mathbf{l}} \\ pred^{\mathbf{p}}(x_2^{\mathbf{v}}, house^{\mathbf{c}}, n.01^{\mathbf{c}})^{\mathbf{l}} \\ pred^{\mathbf{p}}(x_3^{\mathbf{v}}, parliament^{\mathbf{c}}, n.01^{\mathbf{c}})^{\mathbf{l}} \\ pred^{\mathbf{p}}(x_4^{\mathbf{v}}, rally^{\mathbf{c}}, n.01^{\mathbf{c}})^{\mathbf{l}} \\ pred^{\mathbf{p}}(e_1^{\mathbf{v}}, march^{\mathbf{c}}, v.01^{\mathbf{c}})^{\mathbf{l}} \\ named^{\mathbf{p}}(x_5^{\mathbf{v}}, park^{\mathbf{c}}, geo^{\mathbf{c}})^{\mathbf{l}} \\ named^{\mathbf{p}}(x_6^{\mathbf{v}}, hyde^{\mathbf{c}}, geo^{\mathbf{c}})^{\mathbf{l}} \\ role^{\mathbf{p}}(e_1^{\mathbf{v}}, x_1^{\mathbf{v}}, theme^{\mathbf{c}})^{\mathbf{l}} \\ rel^{\mathbf{p}}(x_2^{\mathbf{v}}, x_3^{\mathbf{v}}, of^{\mathbf{c}})^{\mathbf{l}} \\ rel^{\mathbf{p}}(x_5^{\mathbf{v}}, x_6^{\mathbf{v}}, eq^{\mathbf{c}})^{\mathbf{l}} \\ rel^{\mathbf{p}}(e_1^{\mathbf{v}}, x_2^{\mathbf{v}}, from^{\mathbf{c}})^{\mathbf{l}} \\ rel^{\mathbf{p}}(e_1^{\mathbf{v}}, x_4^{\mathbf{v}}, to^{\mathbf{c}})^{\mathbf{l}} \\ rel^{\mathbf{p}}(e_1^{\mathbf{v}}, x_5^{\mathbf{v}}, in^{\mathbf{c}})^{\mathbf{l}} \\ rel^{\mathbf{p}}(e_1^{\mathbf{v}}, t_2^{\mathbf{v}}, \subseteq^{\mathbf{c}})^{\mathbf{l}} \\ rel^{\mathbf{p}}(t_2^{\mathbf{v}}, t_1, \prec^{\mathbf{c}})^{\mathbf{v}} \\ (t_1^{\mathbf{v}} =^{\mathbf{p}} now^{\mathbf{c}})^{\mathbf{l}} \\ \\ \hline \end{array}$$

$\oplus^{\mathbf{p}} (v_1^{\mathbf{v}} @^{\mathbf{p}} e_1^{\mathbf{v}})^{\mathbf{l}})^{\mathbf{t}}$

Note that at sentence-level, rather complex meaning representations are encountered in comparison to lexical logical forms. This permits us to observe patterns in the type-assigned DRSs. For instance, in (6.11), the second argument of all discourse conditions that specify the free-form predicates and their word senses (e.g. $pred(x_2, house, n.01)$) is systematically typed as $constant$. In a similar vein, the last argument of all the conditions that introduce thematic role (e.g. $role(e_1, x_1, theme)$) or spatio-temporal relations (e.g. $rel(x_2, x_3, of)$) is also assigned the same type.

Another related observation is the representation of tense, which is regarded as a sub-lexical phenomenon. As seen in the example, the past tense of the verb is represented using a neo-Davidsonian formulation that involves anchoring of a reified temporality variable to the current point in time (e.g. $t_1 = now$) and specifying the point in time that the event occurred as preceding this reified temporality variable (e.g. $rel(t_2, t_1, \prec)$). Both the anchorage (now) and the time precedence ($\prec$) values that encode the tense are also assigned the $constant$ type.

Document-level logical forms are S-DRS annotations. Discourse partitioning and rhetorical relations are usually encoded at this level, given that documents are composed of multiple sentences. Figure 6.2 is an example document-level logical form annotation and its typed variant. Here, the meaning of the document partitions the discourse structure with a $continuation$ relation between two DRSs. Note that within the scope of the literal that encodes this discourse-level relation ($rel(\pi_1, \pi_2, continuation)$) the differentiating argument that specifies the discourse relation is again typed as $constant$.

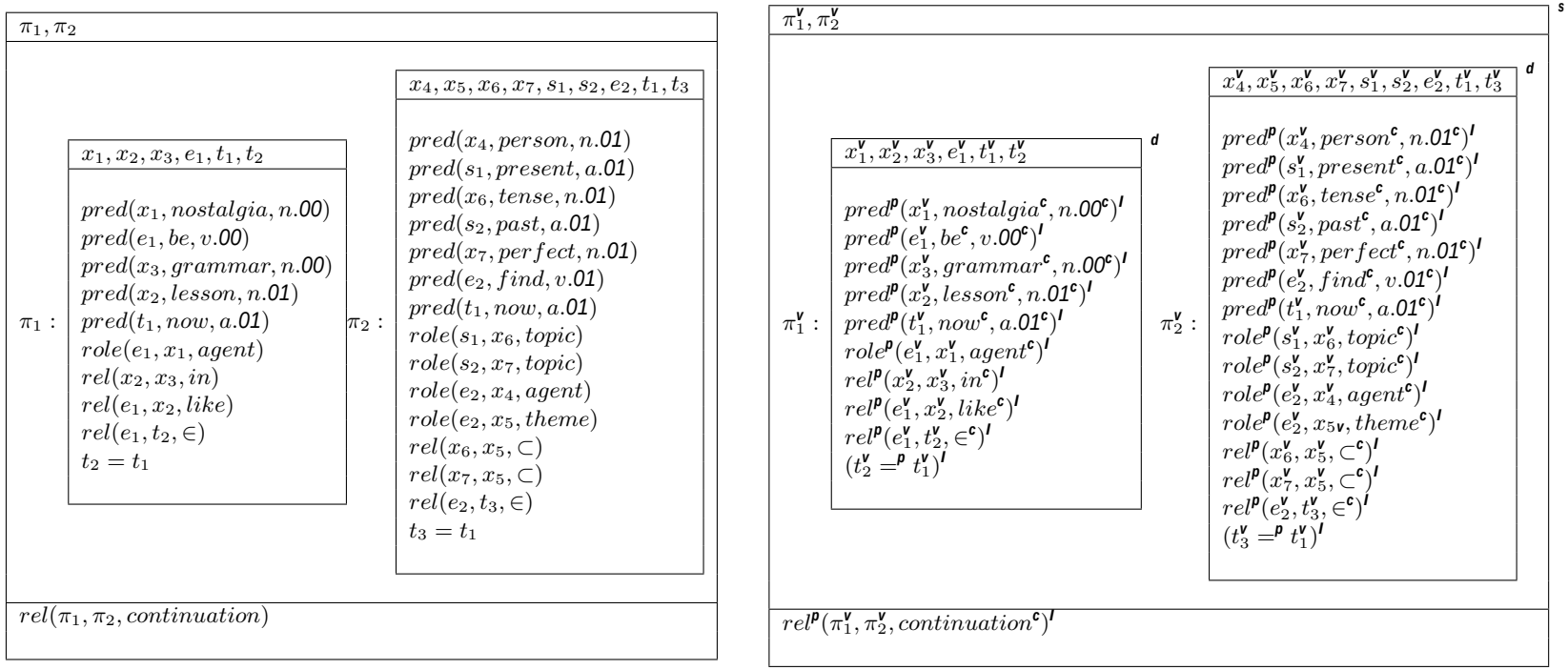


Figure 6.2: (Left) S-DRS annotation for the multi-sentence document '*Nostalgia is like a grammar lesson. You find the present tense and the past perfect.*' from GMB p.15-d.0751. (Right) Type assignment on this S-DRS annotation.

Constant	Open Set?	Examples
Free form	Yes	*bush*, *annual*
Relation	No	*of*, *from*
Discourse relation	No	*continuation*, *parallel*
Temporality	No	*now*
Named entity	No	*per*, *geo*
Thematic role	No	*theme*, *agent*
Year	No	*2014*, *1972*
Month	No	*08*, *03*
Day	No	*12*, *29*

Table 6.1: Classification of constant typed tokens that appear in GMB logical form annotations, together with examples for each class.

Given these observations that span phenomena from sub-lexical level up to discourse, it could be deduced that all the tokens from DRS expressions that are typed $constant$ are special in the sense that they encode the variation in linguistic features across analysis levels. The $constant$ typed tokens are classified as in Table 6.1. These tokens correspond to the values that annotate various phenomena that are conventionally considered disjoint and belong to distinct levels of analysis. Such annotations consist of free-form predicates, spatio-temporal relations, discourse relations, named entity classes, thematic roles, tense, and also date-time expressions. Given the previous definition of the type as a set of values that are functionally equivalent, annotations for these phenomena can now be uniformly processed altogether over type-assigned logical forms as part of task-specific procedures. In the rest of this chapter, such procedures that aim to improve the parsing of natural language text to DRS meaning representations are elaborated.

6.4 Templatisation and Masking of the Logical Form

Introducing types to the logical form facilitates automatic induction of templatic lexical items that encode the lexical meaning as templatised DRS meaning representations, similar to the rules of the GENLEX algorithm that are presented in *Section 5.1, p.56*. This is achieved by employing types in encoding logical forms from sub-lexical level to discourse. The logical form templatisation task is defined as to obtain a templatic logical form, LF^T, for a given DRS meaning representation, LF. Below two concepts are introduced to explain templatisation.

First concept is **full templatisation** of a DRS meaning representation. Consider the

type-assigned parse tree, π, from *Section 6.3, p.72*, which is composed of a set of nodes, $nodes(\pi)$. The fully templatised meaning representation, LF^T, is obtained from this type-assigned parse by replacing all k typed nodes $\forall n_k \in nodes(\pi)$ with a variable, where $k = \textit{constant}$. Second, is the **partial templatisation** of the logical form. In this case, only a subset of the $constant$ typed nodes is templatised with a variable. If N_{const} is the set of $constant$ typed nodes that are templatised, where $N_{const} \subseteq nodes(\pi)$, then $N_{const}^{partial} \subseteq N_{const}^{full}$; hence full templatisation is a special case of partial templatisation.

To illustrate templatisation over an example, consider the lexical logical form annotation in box-notation for the English word '*have*' in (6.12). Here, the set of $constant$ typed tokens, N_{const}, are $\{now, \subseteq, \approx\}$, where these values mark the tense of the verb. In fully templatised form, all these tokens are abstracted with a variable, as in (6.13).

(6.12) $\lambda v_1.\lambda v_2.\lambda v_3.((v_1@v_2)@\lambda v_4.($ $\begin{array}{|l|}\hline t_1, x_1, e_1 \\ \hline \\ rel(e_1, x_1, \subseteq) \\ rel(v_4, e_1, \approx) \\ t_1 = now \\ x_1 = t_1 \\ \\ \hline \end{array}$ $;(v_3@v_4)))$

(6.13) $\lambda v_1.\lambda v_2.\lambda v_3.((v_1@v_2)@\lambda v_4.($ $\begin{array}{|l|}\hline t_1, x_1, e_1 \\ \hline \\ rel(e_1, x_1, c_1) \\ rel(v_4, e_1, c_2) \\ t_1 = c_2 \\ x_1 = t_1 \\ \\ \hline \end{array}$ $;(v_3@v_4)))$

Templatised logical form annotations are similar in form to the semantic types of the rules that are used to bootstrap CCG lexicons with GENLEX, as in *Section 5.1, p.56*. The templatic lexical item that abstracts the lexical logical form annotation of the word '*have*' is shown in (6.14). The fully realised logical form is obtained from this templatic item, if the ordered set of constants $C = \{have, now, \subseteq, \approx\}$ is applied to it. Notice that this templatisation methodology aids in extending the GENLEX lexicon bootstrapping algorithm, and its domain-specific variations, to represent lexical meaning in DRT. Hence, it brings in the full expressive power of DRT to adequately represent the meaning of the open-domain in contrast to the FOPL meaning representations that the previous literature uses in encoding lexical meaning.

(6.14) $\lambda c_1.\lambda c_2.\lambda c_3.\lambda c_4.(c_1 := (\mathsf{S\backslash NP})/(\mathsf{S\backslash NP}) : \lambda v_1\lambda v_2\lambda v_3.((v_1@v_2)@\lambda v_4.($ $\begin{array}{|l|}\hline t_1, x_1, e_1 \\ \hline \\ rel(e_1, x_1, c_1) \\ rel(v_4, e_1, c_2) \\ t_1 = c_2 \\ x_1 = t_1 \\ \\ \hline \end{array}$ $;(v_3@v_4))))$

Templatisation, as introduced here, is closely related to the concept of masking in computational modelling. For example, masking is commonly utilised in pre-training generalised models, which are then fine-tuned on downstream tasks, such as semantic parsing (Devlin et al., 2019; Howard & Ruder, 2018; Peters et al., 2018; Radford et al., 2019). In sequence-to-sequence modelling, the generalised model is pre-trained with the MLM objective. This is an unsupervised learning methodology where the pre-training data is not annotated with labels that mark linguistic phenomena. Using a naive MLM objective, randomly selected tokens of the pre-training examples are masked and the model is tasked with guessing the masked tokens.

In this context, an MLM pre-training objective over DRS logical forms is formalised as a task, where given a partially templatised logical form, LF^T, we want to predict the ordered set of constants, C, that can fully recover the corresponding *LF* after function application. This formalisation of the MLM objective is parallel to the induction of fully realised lexical items from templatic rules in GENLEX as presented above. For instance, the masked lexical logical form annotation for the verb '*have*' after full templatisation is as follows:

(6.15) $\lambda v_1.\lambda v_2.\lambda v_3.((v_1 @ v_2) @ \lambda v_4.($ $\begin{array}{|l|}\hline t_1, x_1, e_1 \\ \hline \\ rel(e_1, x_1, \_\_) \\ rel(v_4, e_1, \_\_) \\ t_1 = \_\_ \\ x_1 = t_1 \\ \\ \hline \end{array}$ $; (v_3 @ v_4)))$

So far, all the examples in this section have illustrated templatisation and masking over lexical meaning representations. However, given that the provided definition of both tasks is generalised over all the mentioned analysis levels, it is possible to extend them to sentence or document-level semantics. To illustrate, Figure 6.3 presents the masked document-level DRS annotation after full templatisation for the type-assigned meaning representation in Figure 6.2. Obtaining the tokenised and linearised form of such masked DRSs annotations yields a set of MLM pre-training examples for a cross-level type-aware pre-training task over document-level logical form annotations.

6.5 Supertagging for Semantic Parsing

The supertagging literature that is reviewed in *Section 3.4, p.24-Section 3.5, p.28* models a syntactic disambiguation task. These models are precursors of a syntactic parser, predicting a beam of syntactic categories, which are used by the parser to construct the derivation for the input (Clark & Curran, 2007). Semantics is then projected onto the disambiguated parse tree as the final step of analysis using a DRS parser, such as Boxer (Bos, 2008). This cascaded analysis pipeline and the involved steps are illustrated in Figure 6.4.

The cascaded analysis has limitations. First, syntactic disambiguation and semantic parse construction are carried out in successive analysis stages, which are not

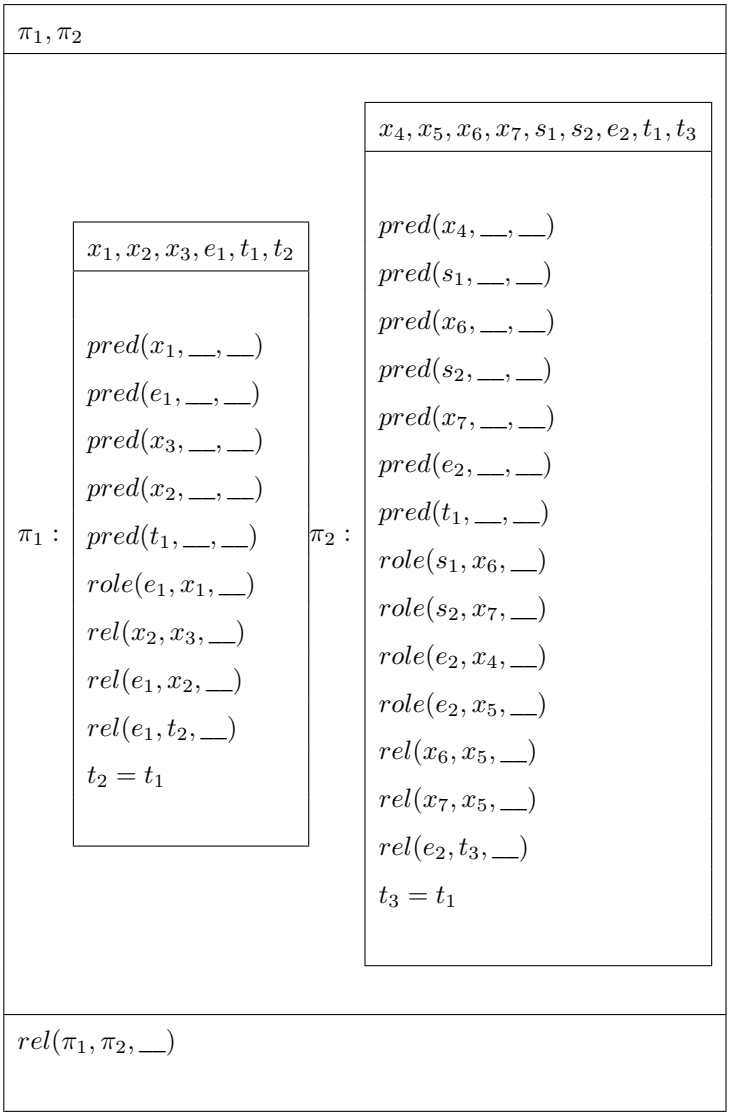


Figure 6.3: Masked form of S-DRS annotation after full templatisation for the multi-sentence document '*Nostalgia is like a grammar lesson. You find the present tense and the past perfect.*' from GMB p.15-d.0751.

conditioned on one another. For instance, semantic ambiguity, which is encoded in the lexicon (e.g. two identical lexical items only differentiated by their semantic type, hence causing lexical ambiguity), does not restrain any disambiguation steps that are taken during supertagging or syntactic parsing. This independent modelling of syntactic and semantic processes does not clearly capture the syntax-semantics interface in obtaining a derivation. Second, the scope of the parse is restricted to standalone sentences. While processing multi-sentence documents with the aim of deriving document-level semantics with discourse structure, first sentence-level logical forms are constructed, which are then merged to obtain the document-level semantics.

Using the logical form templatisation methodology from *Section 6.4, p.76*, it is possible

to extend supertagging to a tagging task that captures syntax-semantics interface at lexicon by predicting a lexical item for each input word instead of only outputting syntactic categories. In that respect, a templatic CCG lexicon, Λ^T, is a set of templatic lexical items, $I^T = \{i_1, \ldots, i_n\}$, where n is the size of the templatic lexicon, $|\Lambda^T|$, and each item $i \in I^T$ has a form similar to (6.14). The supertagging task is then re-formalised as finding an ordered set of 2-tuples, $T = \{t_1, \ldots t_m\}$, for a given sentence, $S = \{w_1, \ldots, w_m\}$, where m is the number of words in the sentence. Here, t_k, where $1 \leq k \leq m$, is the prediction of the supertagger for the corresponding word, w_k, from the input sequence of words. Tuples are in the form, $\langle i, C \rangle$, where $i \in I^T$ is a templatic lexicon item from Λ^T, and C is the ordered set of constants that yield a fully realised lexical item when function applied to i. The size of the predicted set of constants, $|C|$, is equal to the arity of the function defined by i.

Consider the example in Figure 6.5. Here, for the given sentence '*Alice ran the marathon*', the proposed supertagger predicts a sequence of tuples that are composed of templatic lexical items (e.g. for the first word in the input, the predicted item has the syntactic type `NP` and a semantic type that is an λ-abstraction over a DRS with a condition that specifies a templatised named entity), and the set of constants (e.g. $\{alice, alice, per\}$). Given these tuples, fully realised lexical items for the input words can be obtained deterministically through function application. The rest of the analysis procedure is only to derive the compositional interpretation while assuming a set of combinatory rules as in *Section 2.4, p.10*.

The rationale for adopting the proposed approach over the cascaded analysis methodology is two-fold. First, supertagging now not only disambiguates the syntactic categories for the input but also eliminates the lexical ambiguity that arises due to the variation of semantic types among lexical items that are otherwise identical. Second, the approach permits us to model a supertagging task over full multi-sentence documents rather than sentence-based processing while being faithful to the motivation of having a clear representation of the syntax-semantics interface in every step of derivation.

Document-level supertagging is achievable, especially in sequence-to-sequence formalisation of the task, by devising output representations over predicted lexical item templates that subsume the output sequence representation space. For instance, current state-of-the-art Transformer-based pre-trained models that are used in fine-tuning on sequence labelling tasks, such as BERT (Devlin et al., 2019), T5 (Raffel et al., 2020) or their derivatives, have a hard model-specific constrain on the output sequence length.[4] A naive attempt to encode the output sequences of the proposed supertagger using the templatic lexical items in tokenised and linearised form generates sequences that are lengthier than these sequence length constraints when document-level processing is being modelled.

Templatisation of the lexical item in the prediction space makes it possible to encode the predicted templates with an index that maps to an item of an indexed templatic lexicon. In this context, an indexed templatic lexicon, Λ^{Ti}, is defined as a map that is composed of entries in the form $x \rightarrow i$, where x is a unique integer index and i is a templatic lexical item. With this strategy, the predicted tuples have the form $\langle x, C \rangle$,

[4] This constrain usually varies between a maximum of 128 and 1024 tokens for the output token sequence depending on the size of the pre-trained model.

yet this time x is an integer index that selects a templatic lexical item, $\Lambda^{Ti}[x]$, from an indexed templatic lexicon, Λ^{Ti}. This sequence encoding method and similar output representation subsuming approaches make the proposed supertagging task feasible in encoding full document scope as part of the disambiguation task.

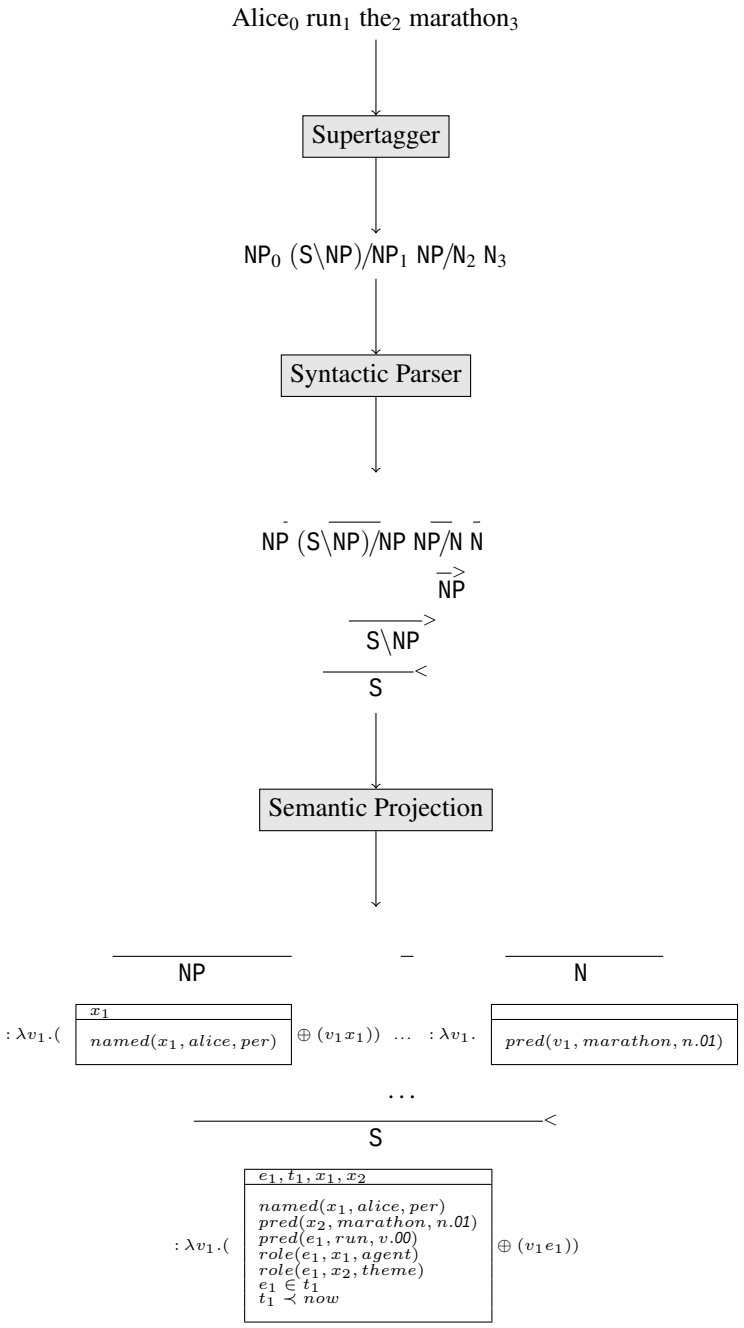


Figure 6.4: Supertagging in a cascaded analysis pipeline first predicts a beam of syntactic categories, which are used to construct a parse tree on which semantics is projected.

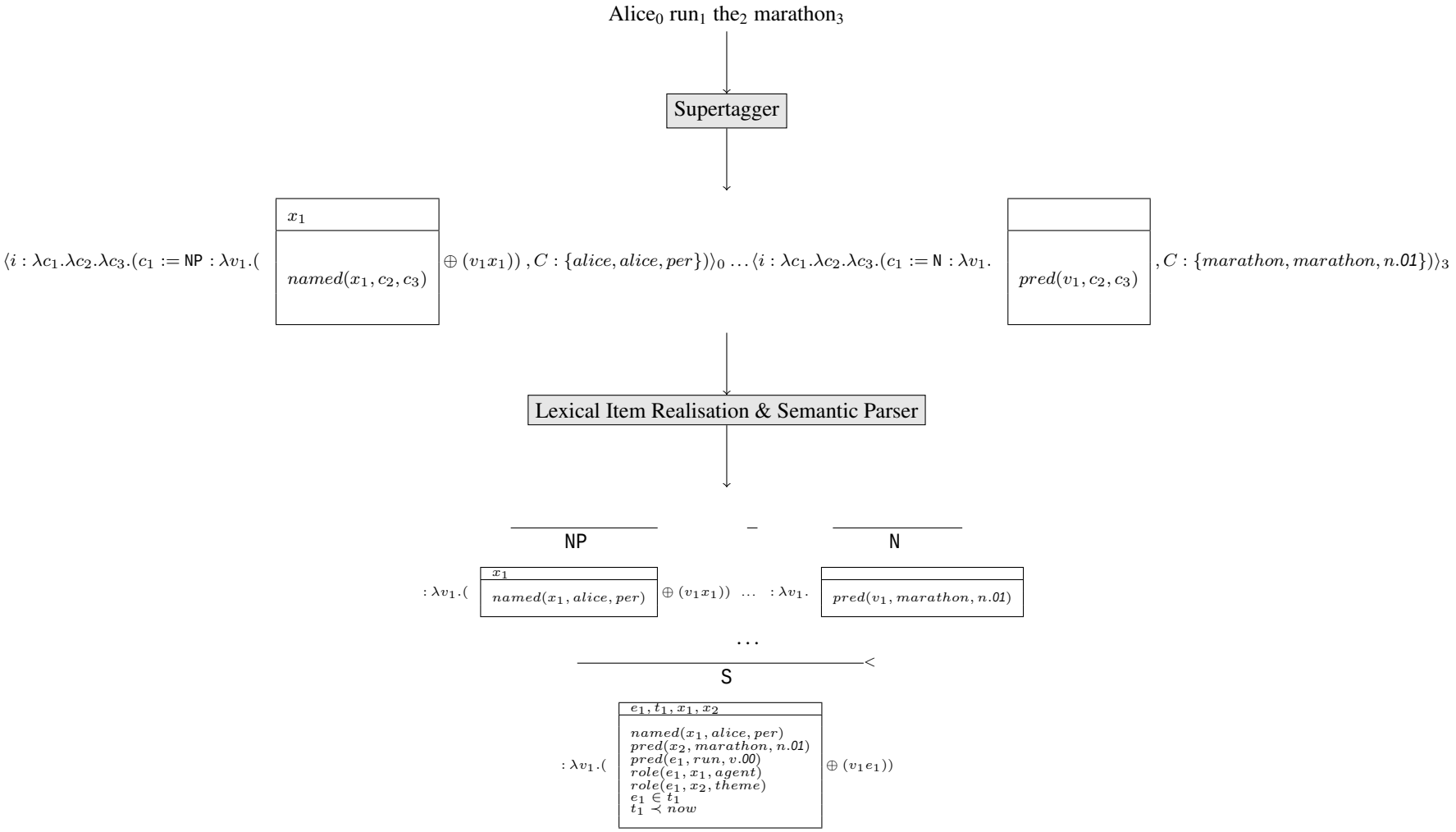


Figure 6.5: Supertagging as a lexical item prediction task, where the output of the supertagger is a tuple of a templatic lexical item and a set of constant values, which are then used to fully realise lexical items to construct a semantic parse.

CHAPTER 7

EXPERIMENTS

This chapter presents the experiments that investigate whether parsing performance on an open-domain semantic parsing problem is improved by embedding input and output sequences using pre-trained representations in an encoder-decoder model. The focus is on DRS parsing, in which open-domain English sentences are mapped to logical forms that constitute DRS meaning representations. In this context, the models that are trained using the Masked Language Modeling (MLM) pre-training objective produce the static embedding vectors that are then used as pre-trained representations in a downstream parsing task. Using the MLM objective from Devlin et al. (2019), such models are pre-trained for the input sequences that are open-domain English sentences. In order to pre-train representations for output DRS meaning representations, the MLM variant, which is derived from the logical form type assignment methodology as presented in *Chapter 6 Cross-Level Typing the Logical Form*, is used.

In *Section 7.1*, the chapter begins by describing the experimental setup and methodology by outlining two sets of experiments that independently test the impact of pre-trained embedding initialisation for input and output sequences. *Section 7.2* presents the encoder-decoder model architecture that is used in downstream DRS parsing. *Section 7.3* proposes a methodology to obtain static embedding vectors from a pre-trained model that outputs contextual representations through distillation. *Section 7.4* introduces a continual training technique that is used to pre-train a model that can learn representations for both the input and output sequences. Finally, *Section 7.5-Section 7.6* reviews the evaluation methodology that is used to assess downstream parser performance and discusses the DRS parsing results with respect to the above mentioned research question.

7.1 Methodology

We present two sets of experiments, which are compared against a baseline to illustrate the effect of type-aware pre-training on DRSs. The baseline is the unaltered Transformer model described in Vaswani et al. (2017). It is trained to generate linearised logical forms for a given GMB sentence.

The experiment models are also based on the Transformer architecture, which are identical to the baseline. However, the weights of the embedding layer of experiment models are initialised using pre-calculated embedding matrices. The embedding weights are distilled from two distinct models:

- **Experiment A:** distillation from a model that is pre-trained on English sentences only.
- **Experiment B**: distillation from a model that is pre-trained on templatised logical forms and also English sentences.

Both the baseline and the experiment models are trained using the identical set of hyper-parameters that are shown in Table 7.1. It is aimed to test variation in parsing performance when embedding layer weights are initialised from pre-calculated values with this configuration. GMB version 2.2.0 is used for all experiments. In all experiments, GMB parts 10–20 are utilised as the development split, parts 00–10 are used as the test split, and the remainder of the SemBank is used as the training split in accordance with the literature. The corresponding development set contains 6,092 examples, test set contains 6,455, where as the training set is composed of 49,175 examples.

The MLM variant shown in *Section 6.4, p.76* can be used for pre-training on DRS logical form annotations of both GMB and PMB. Here, the experiments are carried out on GMB as opposed to PMB for two reasons. First, GMB has a larger set of annotated sentences compared to PMB, despite the fact that it does not guarantee gold annotations, which could impair model performance and generalisation capabilities. van Noord et al. (2020) show that Transformer architecture does not outperform bi-LSTM models on relatively smaller datasets like PMB, while leaving the question of whether the GMB dataset is large enough unresolved. Second, the results obtained through the experiments that are defined above can serve as a baseline together with the findings of J. Liu et al. (2019a) to compare against future research on parsing multi-sentence documents that might derive from the type-aware pre-training objective that is adopted here.

local batch size	16	filter size	2,048
train steps	20,000	hidden dropout	0.1
learning rate	2.0	attention dropout	0.1
decay rate	1.0	relu dropout	0.1
warmup steps	5,000	label smoothing	0.1
initalizer gain	1.0	opt. adam β_1	0.9
hidden size	512	opt. adam β_2	0.997
hidden layers	6	opt. adam ϵ	1e-09
attention heads	8		

Table 7.1: Hyper-parameters that are used in training baseline and experiment Transformer models. We train all models on Cloud TPU chips with 8 cores. The batch size refers to the local batch size (number of sequences per TPU core).

7.2 Parsing Model

7.2.1 Vocabulary

Baseline and experiment models employ WordPiece tokenisation with a vocabulary of 30,522 tokens, similar to Devlin et al. (2019). Both input and target sequences are encoded using this single vocabulary.

In order not to encounter any out-of-vocabulary tokens in training and inference time, the vocabulary needs to include all tokens that appear in tokenised GMB sentences and their corresponding linearised logical forms. To ensure that, the set of all possible tokens that WordPiece tokenisation yields for GMB sentences and their linearised logical forms from all dataset splits are gathered prior to model training as part of a preprocessing stage. This preprocessing begins by assuming a base vocabulary that is inherited from the $\text{BERT}_{\text{base}}$ model. The unused tokens in this base vocabulary are then replaced with those from the previously gathered unified set of all possible tokens and with the set of logical form-specific tokens (such as DRS operators like *lam*, *merge*, or variables such as x_3), only if they are not already present in the base vocabulary.

No special treatment is implemented to represent the possibly open set of variables. The resulting WordPiece vocabulary includes each indexed variable that appears in the dataset. We observe that $\text{BERT}_{\text{base}}$ vocabulary has sufficient unused tokens to accommodate the set of indexed variables that appear in GMB sentence-level logical form annotations.

7.2.2 Sequence Representations

For all models, the input and target sequence representations are identical. Sentences from WordPiece tokenised GMB sentences make up the input sequence. The linearisation algorithm from *Section 6.4, p.76* is applied to the annotated sentence-level logical form of each sentence to generate target sequences. The unique [SEP] token, which signals the end of the sequence for the Transformer's auto-regressive decoding, is appended at the end of every target sequence. Note that [SEP] is also a part of the BERT sequence embedding representation while pre-training with the MLM objective. Therefore, a dual use is assigned to it in the context of our experiments. Below is an example input and output sequence representation for the sentence '*Alice ran a marathon.*':

(7.1) Input: $\{Alice, ran, a, marathon, .\}$
Output: $\{lam(, v_1, drs(, x_1, x_2, named(, ..., app(, v_1, e_1,),), [SEP]\}$

The length of the input and output sequences is determined by the dimensions of the Transformer hidden layers (d_{model}). Examples with input or output sequences that are longer than a *max_seq_length* threshold are excluded from training and evaluation. Table 7.2 presents some experimental *max_seq_length* threshold values together with the portion of the dataset that this constraint discards. In order to retain the size of the

l	Count (%)	l	Count (%)
1024	3 (0.00)	128	54,041 (87.15)
512	**288** (**0.46**)	64	61,738 (99.56)
256	19,034 (30.70)	32	61,996 (99.98)

Table 7.2: The number and percentage of examples that are filtered out for given maximum input and output sequence lengths (l).

Transformer model described in Vaswani et al. (2017), the sequence length threshold is chosen as 512. This enables us to train and evaluate the models with high coverage by keeping the filtered out example set to account for only 0.46% of the GMB sentences. This percentage is computed relative to the totality of 62,010 sentence-level examples in GMB. Across all GMB sentences, the minimum input or output sequence length is 17 and the maximum is 1,340.

7.2.3 Shared Weights

Similar to Press and Wolf (2017), 3-way weight tying is ensured in all models by sharing weights between the embedding layers of the encoder and decoder and the decoder's pre-softmax layer. When the dimensions of the embedding weight vector and the Transformer hidden layers (d_{model}) are different, the outputs of the embedding layer and Transformer decoder block are projected to additional dense layers. In terms of trainable parameters, additional dense layer parameters are the only variation in model architecture between the experiment models and the baseline.

7.3 Feature distillation

In order to obtain static embedding matrices, BERT features are extracted from pre-trained BERT models using a distillation method that is similar to that used by Bommasani et al. (2020). BERT representations are contextual, meaning that BERT does not output the same representation for a token if its context varies in input. Hence, through distillation of contextual representations into static embeddings, it is assumed that the resulting matrix would partially inherit what BERT learns about the structure of its input during pre-training.

Let S=$\{S_1, S_2, ..., S_i\}$ be the set of sequences. In all experiments, S is composed of sequences from the training split examples of GMB. Let W be the set of all vocabulary tokens and W_S be the set of unique tokens that appear in S, where $W_S \subset W$. The aim is to induce a $n \times m$ static embedding matrix E, where n is the size of the vocabulary $|W|$, and m is the dimension of the hidden layers of the BERT model.

For a sequence S_k=$(w_1, w_2, ..., w_l)$, of length l, $k \leq i$, where i is the number of

sequences, BERT outputs vectors H_k=$(h_1, h_2, ..., h_j)$ where h_j is the vector of representations from the last hidden layer. H_k is pooled into a $l \times m$ feature vector f using the following pooling strategies:

(7.2) **First layer:** $f = h_1$
Penultimate layer: $f = h_{j-1}$
Last layer: $f = h_j$
Sum of last four layers: $f = sum(h_{j-3}, ..., h_j)$
Sum of all layers: $f = sum(H_j)$

By pooling BERT representations for each input sequence in S a set of feature vectors F=$\{f_1, f_2, ..., f_i\}$ is obtained. For the token w_k^i, $i \leq l$, the corresponding feature instance f_k^i can be gathered, which is a 1-dimensional vector with length m. Such feature instances for $\forall w \in W_S$ are then aggregated and their arithmetic mean is taken to obtain the distilled static embedding for that token.

For tokens that appear in the sequences, $\forall w \in W_S$, the distilled embedding vector is normalised using min-max normalisation to scale its values between $[-x, x]$ where $x = \sqrt{d_{model}}$. For all other tokens, $\forall w \in W, w \notin W_S$, random values from a normal distribution that has a mean of 0 and a standard deviation $1/\sqrt{d_{model}}$ is used as the embedding. This randomisation is also utilised in the initialisation of the embedding layer weights for all tokens in the baseline model. The assumption is that this normalisation helps to avoid Transformer bias towards extracted static token embeddings over randomly initialised ones since Vaswani et al. (2017) multiply embedding weights with $\sqrt{d_{model}}$ prior to the positional encoding.

7.4 Continual Pre-training

For the first set of experiments (Experiment A) features are only extracted for the WordPiece tokens that appear in input (tokenised GMB sentences) using the $\text{BERT}_{\text{base}}$ checkpoint. Thus, in this set of experiments, the embedding weights are pre-computed only for the tokens that are input to the Transformer encoder block, whereas they are randomly initialised for the tokens that only appear in linearised logical forms.

For the second set of experiments (Experiment B) a continually pre-trained BERT model is employed. To obtain this model, we start with the $\text{BERT}_{\text{base}}$ checkpoint and continue pre-training it with the MLM objective variant that is derived from the logical form type assignment. For continual pre-training, the examples from the training split of GMB and the parameters from Table 7.3 are used. This time, BERT representations are extracted for the tokens of both input and target sequences from this model.

For each logical form, 10 pre-training examples are generated and 15% of the *constant* typed tokens, or at most 20 tokens per example, are masked. The same masking strategy that Devlin et al. (2019) employs is used; 10% of the time the token that is chosen as a candidate is replaced with a random token from the vocabulary, 10% of the time the candidate is kept as-is, and for the rest of the time it is replaced with a special masking symbol ([MASK]). If *constant* typed tokens are split due to WordPiece tokenisation, sub-tokens of a constant are allowed to be masked independently.

local batch size	16	hidden layers	12
train steps	125,000	attention heads	12
learning rate	5e-05	hidden dropout	0.1
warmup steps	25,000	attention dropout	0.1
hidden size	768		

Table 7.3: Hyper-parameters that are used in continually pre-training $\text{BERT}_{\text{base}}$ model. We pre-train on Cloud TPU chips with 8 cores.

The resultant pre-training examples are therefore partially templatised sentence-level meaning representations.

7.5 Evaluation

There are two algorithms proposed in the literature to evaluate DRS parsers, **Counter** (van Noord, Abzianidze, Haagsma, et al., 2018) and **Dscorer** (J. Liu et al., 2020).

Counter is the standardised cross-language evaluation method as it was used to provide parsing metrics on PMB in *The First Shared Task on DRS Parsing* (Abzianidze et al., 2019). To evaluate with Counter each ground-truth and predicted DRS pair in the evaluation set is mapped to clausal form representations, which is presented in *Section 6.1, p.68*, by flattening the recursive DRS. The clauses mark the membership relations of discourse referents and conditions to corresponding DRSs and also the argument values of predicates such as DRS operators, thematic roles, word senses (e.g. the clause 'b_1 *REF* x_1' translates to 'variable x_1 is a referent of DRS indexed as b_1'). Counter computes the maximal one-to-one mapping between a set of clauses generated using hill-climbing. In this sense, it is a derivation of SMATCH that is used to evaluate AMR parsers (S. Cai & Knight, 2013).

On the other hand, Dscorer first induces graph representations for DRS pairs, where sub-categorised variables of a clause are mapped to graph nodes. Clause operators and their values are projected to the edges between such graph nodes. The similarity of a DRS pair is obtained by computing the matching n-grams that are extracted for each node.

We use Counter from van Noord, Abzianidze, Haagsma, et al. (2018) to obtain the results that are presented in Table 7.4. Dscorer is a computationally efficient evaluation algorithm compared to Counter for DRSs which yield graph representations that are larger in size. Yet, the computational cost of the evaluation is not prioritised since the evaluation is carried out on comparably smaller sentence-level meaning representations.

		Test Set				Dev Set			
		Ill	P	R	F1	Ill	P	R	F1
Baseline		1.34	76.17	83.49	79.66	1.82	75.84	83.59	79.52
$BERT_{base}$	first	1.48	77.39	83.26	80.22	1.73	77.08	83.39	80.11
	penultimate	1.79	76.25	83.50	79.70	2.04	75.95	83.47	79.54
	last	1.28	77.37	83.27	80.21	1.56	77.16	83.39	80.15
	sum of last 4	1.48	76.67	83.41	79.90	1.70	76.51	***83.58***	79.89
	sum of all	1.32	77.38	83.35	80.25	1.50	77.66	83.49	80.47
Continually pre-trained BERT	first	1.24	77.67	83.04	80.26	1.42	77.46	83.26	80.26
	penultimate	1.88	76.44	83.39	79.75	2.10	76.01	83.50	79.57
	last	1.06	***78.26***	83.13	***80.62***	1.27	***78.01***	83.10	***80.48***
	sum of last 4	1.14	77.23	83.35	79.86	1.28	76.62	83.42	79.88
	sum of all	***0.94***	77.47	***83.51***	80.37	***1.26***	77.11	83.45	80.15

Table 7.4: Percentage of ill-formed logical form expressions produced (Ill), precision (P), recall (R) and F1-score for the baseline and experiment models on GMB test and development sets. All results are averaged over 3 model training and evaluation runs per experiment case.

Note that the clausal form only captures (S-)DRS scoped meaning. Hence, during clausal form conversion lambda/ expressions, DRS operators (such as $merge$ and $apply$) and their variable arguments from predictions are discarded. These tokens essentially act as the glue logic to compose multi-sentence semantics.

Prior to evaluation with Counter **DRS-JURY** (van Noord et al., 2020) is used to filter out structurally ill-formed predicted logical forms and the evaluation metrics are computed only over the set of examples for which the model outputs well-formed predictions. DRS-JURY provides clausal form signatures that can recognise the structure of PMB annotations, but not GMB semantic representations. The set of DRS-JURY signatures is adapted to parse and validate the representation of GMB logical form annotations.

7.6 Discussion

The results from Table 7.4 are discussed below with respect to the two controlled variables in the experiments, which are pre-training and feature distillation.

Does pre-trained representations improve parsing performance?

The results illustrate that the pre-trained representations improve DRS parsing performance. Initialising the embedding layer weights of a Transformer model, regardless of the employed pre-training and feature distillation strategy, improves DRS parsing accuracy. All models from both experiment sets have higher F1-scores compared to the baseline. The highest gain is +0.96 for the test set and +0.95 for the development set.

The results also confirm the findings of van Noord et al. (2020). When pre-trained representations are used to embed natural language input, the Transformer model performs better than random initialisation of embedding layer weights for input sequences. Best input representations are obtained from $\text{BERT}_{\text{base}}$ using *sum of all layers* distillation.

It is observed that the best performing model is obtained from pre-computed embedding weights that are distilled with the *last layer* strategy for both natural language input and logical form sequences from a $\text{BERT}_{\text{base}}$ model that is continually pre-trained using the MLM variant over typed-DRS expressions. It is notable that this model yields a +2.09 and + 2.17 precision gain on the test and development sets. This model also results in a 29.85% reduction in the ill-formed DRSs that it predicts compared to the baseline.

Does embedding output with representations from a model that is pre-trained with MLM over typed-DRSs improve DRS parsing?

The model performance improvement when embedding output sequences with pre-trained representations depends on the distillation strategy when decontextualised static embeddings are employed.

When the performance of models from both sets of experiments is compared with

respect to the distillation strategy, continually pre-trained models are shown to yield better representations and model performance. The only exceptions are the *sum of all layers* and the *sum of last 4 layers* strategies. It can be hypothesised that this is due to the update of hidden layer weights via back-propagation during continual pre-training. What is learned from the masking of typed-DRSs is accumulated in the last hidden layer of the model. The neutral performance change in continual pre-training over $\text{BERT}_{\text{base}}$ using *penultimate layer* distillation supports this assumption.

Focusing on the *last layer* distillation, compared to the case where only natural language input is embedded with representations from $\text{BERT}_{\text{base}}$, continual pre-training on typed-DRSs and embedding both natural language and logical form sequences with distilled representations yields further improvements in F1-score (+0.41 and +0.33 for the test and development sets) and precision (+0.89 and +0.85 for the test and development sets).

Which distillation strategy is favourable?

The choice of distillation strategy depends on the pre-training scheme. The results suggest that obtaining features from the last layer of BERT to embed natural language input in DRS parsing might not be the best strategy, while *sum of all layers* distillation yields the best representations for natural language input. This aligns with the findings from Tenney et al. (2019) on what BERT learns about the syntax and semantics of natural language is distributed across its hidden layers.

In the case of distilling logical form representations, it can be hypothesised that the simple yet effective continual pre-training methodology favours the *last layer* distillation. It could be assumed that a method which incorporates masked modelling over typed-DRSs in pre-training BERT from scratch might also result in distribution of learned logical form structural knowledge across the hidden layers of this encoder.

CHAPTER 8

CONCLUSION

This thesis presented a methodology to assign types to meaning representations in order to capture cross-level phenomena as a set of values that can be processed jointly in procedures that aim to improve semantic parsing of natural language text input to logical form representations.

The linguistic theory that was adopted to carry out the analysis was Combinatory Categorial Grammar (CCG; Steedman, 1993, 1996, 2000). *Chapter 2 Combinatory Categorial Grammar* presented the lexicalised nature of CCG, the syntax-semantics interface at the lexicon, and showed the process of compositional derivation of interpretation while assuming principles of categorial government and adjacency. In progression to the logical form type assignment, *Chapter 4 Representing Open-Domain Meaning* illustrated that the semantic types can be encoded using meaning representations from Discourse Representation Theory (DRT; Kamp and Reyle, 1993) as part of lexical item definitions together with neo-Davidsonian event and temporality semantics, which permits us to jointly capture phenomena that are observed in document-level open-domain meaning, such as tense, word sense, thematic role, named entity classes, and discourse relations.

In *Chapter 6 Cross-Level Typing the Logical Form*, the logical form type assignment methodology was introduced as an extension of shift-reduce parsing of the expressions that are obtained by linearising the structured meaning representations. The type system was derived from a grammar that defines the syntax of the logical form language. Both the type system induction and cross-level type assignment were illustrated using Discourse Representation Structures (DRS), as they are annotated in Groningen Meaning Bank (GMB; Bos et al., 2017) and Parallel Meaning Bank (PMB; Abzianidze et al., 2017). The resulting typed DRS meaning representations were shown to uniformly encode above mentioned phenomena across levels as values of the same type.

The approach that is presented in this thesis has various implications for linguistic analysis. Given the transparency of syntax to semantics in CCG, compositionality is ensured, similar to the Montagovian approach to grammar (Montague & Thomason, 1975). That is to say, syntax dictates the order in which surface form constituents combine, while interpretation of adjacent constituents is unified to obtain the derived and reduced predicate-argument structure (Steedman, 1996). Previous literature on CCG that outlines the assumptions of the theory in explaining the disordering of surface structure phenomena, such as bounded and unbounded constructions, with respect to the objects of interpretation, and also the computational modelling research

on bootstrapping semantic parsers, employ meaning representations that are based on first-order predicate logic (FOPL). Together with the mechanics of unification in hand, FOPL representations of interpretation can capture the underlying process that yields the predicate-argument structure. Baseline tools that λ-calculus provides, such as abstractions and well-defined procedure of application, help formalise an adequate representation of scoped meaning as part of the interpretation. Although, such FOPL meaning representations are restricted in the sense that semantic phenomena that we can afford to represent are limited to the scope of a sentence.

DRT, as a dynamic semantics theory, helps alleviate this restriction and can capture meaning beyond the scope of standalone sentences by providing the mechanisms for merging elaborate representations of decoupled sets of discourse referents together with the predicate-argument structure. The assumptions of the theory that are based on accessibility constraints provide an explanation for capturing and resolving coreferentiality relations. Moreover, segmented and projective extensions increase the expressivity of miminal meaning-bearing units of the theory to capture discourse-level phenomena, such as rhetorical relations, and the anaphoric nature of presuppositions. Compositionaly is introduced to DRT by formalising DRS as part of a λ-calculus language (Bos, 2003, 2008, 2009b). Within the context of CCG, the switch from FOPL representations of interpretation to DRT translates to a modification of the language that is used to represent interpretation as part of the semantic type of lexical items. Resources, such as GMB and PMB, provide the empirical data that demonstrates the applicability of such an approach in representing the meaning of open-domain and wide-coverage natural language text in a set of typologically diverse languages.

The focus of this thesis was on such expressive compositional meaning representations that capture phenomena across analysis levels, from sub-lexical level (e.g. tense) to discourse (e.g. rhetorical relations). The thesis showed that linguistic features that are traditionally analysed disjointly can be encoded as part of the representational language that makes up the semantic type of lexical items in a lexicalised grammar theory. Typing of the semantic values in lexical specifications allows us to uniformly represent cross-level phenomena as instantiations of values that belong to the same typed class. Compositionality ensures that any type-dependent procedure that we introduce in the lexicon will also manifest itself in higher-level interpretation after unification through derivation. Moreover, since the type assignment procedure that this thesis describes is self-contained in terms of only being derived from the syntax of the logical form language, the methodology is transparent to phonological and syntactic typological differences.

Besides the lingustic analysis, the thesis also showed some of the implications of logical form typing for semantic parsing in *Chapter 6 Cross-Level Typing the Logical Form*. First, DRS logical form templatisation algorithms are defined, which are then used to illustrate the close correspondence between typing and masking from Masked Language Modelling (MLM; Devlin et al., 2019). Second, the typed DRS meaning representations were used to encode templatic lexical items in order to subsume the output representation space as part of an extension to supertagging (Bangalore & Joshi, 1999), which eliminates the necessity to cascade various analysis levels in computational modelling.

One of the applications of the presented logical form type assignment technique on

computational modelling of parsing is also empirically tested in a series of experiments in *Chapter 7 Experiments*. The scope of the problem in these experiments was to test the effect of the initialisation of the embedding layer of an encoder-decoder semantic parsing model using pre-trained static embedding vectors against the case where the embedding layer is randomly initialised. The pre-trained embeddings were obtained from a generalised model, which was first pre-trained on English sentences only, and then on both sentence-level logical forms and English sentences as part of a separate test case. In both cases, a MLM pre-training objective was employed in pre-training, where in the latter test case the logical forms are masked using a logical form templatisation method that is based on the logical form type assignment procedure that this thesis presents.

The variation of parsing performance for all experiment models was tested in a DRS parsing task, where English sentences from GMB were input to the model and DRS logical form representations were predicted as the output. The experiment results illustrate that initialisation of the parser embedding weights with vectors that are distilled from a model that is pre-trained on English sentences alone improves parsing performance in terms of F1-score compared to the case of random initialisation of the embedding layer. Moreover, the experiment results have also shown that further parsing performance gains are possible when embeddings are distilled from a model that is pre-trained on templatised DRS logical forms.

8.1 Limitations

The limitations of this thesis are discussed below in terms of the generality of the type assignment methodology across representation formalisms, the data that is used in developing the type system and the experiments, and the model of choice to carry out downstream semantic parsing experiments.

Generality of the Approach. In *Chapter 6 Cross-Level Typing the Logical Form* the logical form type assignment methodology is introduced only for typing the DRS meaning representations. In order to assess the generality of the presented approach, further investigation into the applicability of the typing algorithm to formalisms other than DRT is needed.

The base requirement for such an investigation is that representations should be well-formed grammatical expressions of a well-defined logical form language. If the representation in hand accommodates ad-hoc one-off annotation mechanisms to capture edge phenomena that are observed in the data, it might not be possible to craft a grammar that recognises those logical form annotations. For instance, it is trivial to define a context-free grammar for the first-order predicate logic (FOPL) representations that are commonly adopted by the previous literature which are composed of quantifiers, variables, predicates and λ-abstractions (such as logical forms of Zettlemoyer and Collins (2005, 2007)), which can be used to induce a type system.

Although, extensions to such FOPL representations would not be the most interesting exercise given that they lack to encode cross-level phenomena. Some examples adopt neo-Davidsonian representations (Reddy et al., 2014) and provide encoding of tense

and sentence-level phenomena with varying degrees of granularity. However, they do not encode any phenomena beyond the sentence scope, such as discourse relations. Interesting candidates are the Abstract Meaning Representations (Banarescu et al., 2013) and Universal Conceptual Cognitive Annotation (Abend & Rappoport, 2013), which encode rich linguistic information in graphs that can be linearised to work on as expressions of uniform representation languages. Also, to study the domain-specificity of the proposed typing approach, logical forms that are structured executable representations, such as tables (Herzig et al., 2020) or SQL expressions (Zhong et al., 2017) that can be queried against a knowledge base and the slot-filling style meaning representations (Dahl et al., 1994; Iyer et al., 2017; Yu et al., 2018; Zelle & Mooney, 1996) that are crafted for dialogue systems are also of interest. The thesis does not work out the applicability of type assignment to these formalisms, which is essential to claim generality.

Dataset. As it is true for all research that uses machine learning, the quality of the data has a direct impact on the empirical results. The results of the experiments in this thesis are no exception. Both the type assignment methodology and the sequence representations in experiments are designed to capture much of the information that is encoded in the annotations of the dataset that we are working on, which is the GMB. The annotations of GMB are semi-gold, meaning that not all logical form annotations are gold standard. Even though, a portion of the automatically bootstrapped annotations of the dataset are corrected by human experts, errors still exist in all the annotations layers, from syntactic categories to linguistic phenomena that are annotated as part of the logical forms.

One observation is that the CCG that is used to annotate GMB exhibit a high degree of ambiguity. The time cost of semantic parsing sentences from the meaning bank using a CYK parser that has a packed forest chart implementation together with utilisation of all known normal-form rules is impractically high, which is an indication of the ambiguity that the lexical annotations result in. These annotation errors potentially harm the models' ability to generalise over underlying patterns when linguistic phenomena that are observed on the surface form are not annotated regularly across all examples. Without a dataset with gold standard annotations, it is not possible to empirically assess the extent of model quality degradation due to annotation errors. Human-in-the-loop style annotation correction cycles are tedious and time-consuming. However, novel semi-automated techniques that might be devised for improving the annotation quality of the dataset can help to investigate the error margin caused by annotation errors.

Experiment Model. The parsing model that is used in *Chapter 7 Experiments* is an unaltered Transformer model. The intention of using this parsing model without any task-specific architectural modifications is two-fold. First, the motivation of the experiments is to test the effect of type-aware pre-training on warm starting embedding weights in isolation without mixing in the contributions of parsing improvements gained from model architecture modifications. Second, the pre-training and parsing models are chosen to be one of the state-of-the-art sequence-to-sequence models so that the results can be replicated and cross-compared across tasks that use similar pre-training objectives.

This approach ensures that the controlled variables of the experiments are tested standalone, but it does not guarantee that the model architecture is the best performing

choice for DRS parsing task. Pre-trained and distilled embeddings can as well be used to initialise embedding layers of other encoder-decoder models, such as bi-LSTMs without 3-way weight tying, which helps to initialise input and output embedding layers disjointly in contrast to the experiments in this thesis. Also, it has been shown in the literature that copy mechanisms (J. Liu et al., 2018) or type-dependent decoding (Muller et al., 2012) that are implemented as part of the model architecture improve parsing accuracy. Such task-specific architecture improvements can be further studied to examine their joint effect on parsing accuracy together with pre-trained embedding initialisation. Finally, the results may also improve after extensive fine-tuning of the hyper-parameters of both the pre-training and parsing models, which was not carried out in the experiments.

8.2 Future Work

The type assignment methodology and the experiments that are presented in this thesis lay the ground for potential future work in below outlined lines of research.

Document Parsing. The literature on DRS parsing focuses on sentence-level processing. One exception to this is the research of J. Liu et al. (2019a), where a multi-layer LSTM is used to decode DRS output in parsing of full documents, which presents the first empirical results on document-level processing.

The leap forward to making multi-sentence documents the unit of parsing is hindered by two factors. First, most of the state-of-the-art models formalise DRS parsing as a sequence-to-sequence translation problem. Contemporary sequence-to-sequence models have sequence length limitations that are well below the lengths of the sequences that are obtained by naive linearisation of document-level DRS logical forms. Secondly, such models do not take into account compositionality and consider syntactic structure as a latent variable that is learned from surface form.

In *Chapter 6 Cross-Level Typing the Logical Form*, this thesis presents a recipe to subsume the output sequence representation by introducing a reformulation of supertagging to predict lexical items, rather than syntactic categories. Such methods that are directly derived from typing the logical form exhibit the potential to obtain compact output sequence representations and also to reintroduce compositionality in parsing while attaining sequence-to-sequence formalisation. Results from experiments on parsing documents using the supertagger extension or other novel methods that are derived from typing the logical form over its syntactic structure are essential to examine inference of phenomena that are observed beyond the scope of the sentence.

Multilingual Parsing. The type assignment methodology from *Chapter 6 Cross-Level Typing the Logical Form* is transferable to PMB as well. Similar to GMB, PMB also annotates logical forms using DRT. Therefore, the grammar defined for this logical form language in this thesis can be adapted to recognise the annotations of PMB with minor modification. The experiments were only run on GMB, so the experimental results of this thesis are monolingual while only demonstrating the use of pre-trained embeddings in parsing sentences of English. Given that PMB is a multilingual resource, the experiments from *Chapter 7 Experiments* can be run for all the 4 languages that

PMB covers. This would help to carry out a multilingual analysis of DRS parsing on typologically different languages using a typing technique on the output meaning representation that is agnostic to the language that is input to the parser.

In terms of dataset augmentation, the type assigned logical form templatisation technique can be used to expand both GMB and PMB to languages that are not covered by these resources. For instance, machine translation-based methods, such as the slot-filler model that is presented in Nicosia et al. (2021), aim to fill in the slot values of an underspecified logical form using spans that are extracted from a translation of the input sentence. The underspecified logical form in these approaches can be obtained using the templatisation methodology that is outlined in *Chapter 6 Cross-Level Typing the Logical Form*. For a given sentence or document from GMB or PMB, the translation of the input can be obtained in target languages using machine translation models, and the templatic logical form could be used as the pivot representation whose constant values will be filled using surface form spans from those translations. In addition to the alignment-based method that is used by Abzianidze et al. (2017) to create PMB, machine translation-based methods could help scale augmentation of DRT meaning banks.

Mapping Spoken Language to Logical Forms. This thesis makes the assumption in *Chapter 2 Combinatory Categorial Grammar* that all input to the computational models is in written form. Hence, the input to the experiment parser is natural language text, and the models presented in this thesis are learning from supervision on orthographic surface form. Experimenting with parsing spoken language to logical forms can help to alleviate this assumption. In the case of DRS parsing, the textual input of GMB and PMB can be augmented to represent spoken modality in future work.

The sentences in these resources can be transliterated to IPA using a pre-trained monolingual transliteration model, where the words of each sentence are contextually mapped to the corresponding phonetic representation. This approach would not require any alignment between the logical form and the IPA representation of the input, since logical form annotations do not contain any references to any span in the textual representation of the input. The phonetic representation of the natural language input enables experimentation on either end-to-end modelling of parsing IPA representations into logical forms or on joint encoding of the phonetic and orthographic representations using a dual-encoder model architecture that is similar to the model that van Noord et al. (2019) present to encode syntactic features together with the orthographic surface form.

REFERENCES

Abend, O., & Rappoport, A. (2013). Universal Conceptual Cognitive Annotation (UCCA). *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 228–238. https://aclanthology.org/P13-1023

Abzianidze, L., Bjerva, J., Evang, K., Haagsma, H., van Noord, R., Ludmann, P., Nguyen, D.-D., & Bos, J. (2017). The Parallel Meaning Bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 242–247. https://aclanthology.org/E17-2039

Abzianidze, L., & Bos, J. (2019). Thirty musts for meaning banking. *Proceedings of the 1st International Workshop on Designing Meaning Representations*, 15–27. https://doi.org/10.18653/v1/W19-3302

Abzianidze, L., van Noord, R., Haagsma, H., & Bos, J. (2019). The first shared task on Discourse Representation Structure parsing. *Proceedings of the IWCS Shared Task on Semantic Parsing*. https://doi.org/10.18653/v1/W19-1201

Artzi, Y., Das, D., & Petrov, S. (2014). Learning compact lexicons for CCG semantic parsing. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1273–1283. https://doi.org/10.3115/v1/D14-1134

Artzi, Y., & Zettlemoyer, L. (2011). Bootstrapping semantic parsers from conversations. *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 421–432. https://www.aclweb.org/anthology/D11-1039

Artzi, Y., & Zettlemoyer, L. (2013). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, *1*, 49–62. https://doi.org/10.1162/tacl_a_00209

Asher, N. (1993). *Reference to Abstract Objects in Discourse*. Kluwer Academic Publishers.

Asher, N., & Lascarides, A. (2003). *Logics of Conversation*. Cambridge University Press.

Baldridge, J. (2002). *Lexically Specified Derivational Control in Combinatory Categorial Grammar* (Doctoral dissertation). University of Edinburgh.

Baldridge, J., & Kruijff, G.-J. M. (2003). Multi-modal Combinatory Categorial Grammar. *10th Conference of the European Chapter of the Association for Computational Linguistics*. https://aclanthology.org/E03-1036.pdf

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., & Schneider, N. (2013). Abstract Meaning Representation for sembanking. *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, 178–186. https://aclanthology.org/W13-2322.pdf

Bangalore, S., & Joshi, A. (1999). Supertagging: An approach to almost parsing. *Computational Linguistics*, *25*(2), 237–265. https://aclanthology.org/J99-2004.pdf

Barendregt, H. P. (1984). The lambda calculus: Its syntax and semantics. *Studies in Logic and the Foundations of Mathematics*, *103*.

Bar-Hillel, Y. (1953). A quasi-arithmetical notation for syntactic description. *Language*, *29*(1), 47–58.

Basile, V., Bos, J., Evang, K., & Venhuizen, N. (2012a). Developing a large semantically annotated corpus. *Proceedings of the 8th International Conference on Language Resources and Evaluation*, 3196–3200. http://www.lrec-conf.org/proceedings/lrec2012/pdf/534_Paper.pdf

Basile, V., Bos, J., Evang, K., & Venhuizen, N. (2012b). A platform for collaborative semantic annotation. *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, 92–96. https://aclanthology.org/E12-2019

Bhargava, A., & Penn, G. (2020). Supertagging with CCG primitives. *Proceedings of the 5th Workshop on Representation Learning for NLP*, 194–204. https://doi.org/10.18653/v1/2020.repl4nlp-1.23

Bjerva, J. (2014). Multi-class Animacy classification with semantic features. *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, 65–75. https://doi.org/10.3115/v1/E14-3008

Blackburn, P., & Bos, J. (2005). *Representation and Inference for Natural Language. A First Course in Computational Semantics*. CSLI.

Bodenstab, N., Dunlop, A., Hall, K., & Roark, B. (2011). Beam-width prediction for efficient context-free parsing. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 440–449. https://aclanthology.org/P11-1045

Bommasani, R., Davis, K., & Cardie, C. (2020). Interpreting pretrained contextualized representations via reductions to static embeddings. *Proceedings of the 58th*

Annual Meeting of the Association for Computational Linguistics, 4758–4781. https://doi.org/10.18653/v1/2020.acl-main.431

Bos, J. (2003). Implementing the binding and accommodation theory for anaphora resolution and presupposition projection. *Computational Linguistics*, *29*(2), 179–210. https://aclanthology.org/J03-2002.pdf

Bos, J. (2008). Wide-coverage semantic analysis with Boxer. *Semantics in Text Processing 2008 Conference Proceedings*, 277–286. https://www.aclweb.org/anthology/W08-2222.pdf

Bos, J. (2009a). Economical Discourse Representation Theory. *International Workshop on Controlled Natural Language*, 121–134.

Bos, J. (2009b). Towards a large-scale formal semantic lexicon for text processing. *Proceedings of the Biennal GSCL Conference, From Form to Meaning: Processing Texts Automatically*, 3–14.

Bos, J. (2015). Open-domain semantic parsing with Boxer. *Proceedings of the 20th Nordic Conference of Computational Linguistics*, 301–304. https://aclanthology.org/W15-1841.pdf

Bos, J. (2016). Expressive power of Abstract Meaning Representations. *Computational Linguistics*, *42*(3), 527–535. https://aclanthology.org/J16-3006.pdf

Bos, J., Basile, V., Evang, K., Venhuizen, N. J., & Bjerva, J. (2017). The Groningen Meaning Bank. *Handbook of linguistic annotation* (pp. 463–496). Springer.

Bozşahin, C. (2002). The combinatory morphemic lexicon. *Computational Linguistics*, *28*(2), 145–186. https://doi.org/10.1162/089120102760173634

Bozşahin, C., & Güven, A. B. (2018). Paracompositionality, mwes and argument substitution. *International Conference on Formal Grammar*, 16–36.

Cai, Q., & Yates, A. (2013). Large-scale semantic parsing via schema matching and lexicon extension. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 423–433.

Cai, S., & Knight, K. (2013). Smatch: An evaluation metric for semantic feature structures. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 748–752. https://aclanthology.org/P13-2131

Chen, J., & Vijay-Shanker, K. (2000). Automated extraction of TAGs from the Penn Treebank. *Proceedings of the 6th International Workshop on Parsing Technologies*, 65–76. https://aclanthology.org/2000.iwpt-1.9

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, *2*(3), 113–124.

Chomsky, N. (1981). *Lectures on Government and Binding*. Foris Publications.

Christodouloupoulos, C., & Steedman, M. (2015). A massively parallel corpus: The Bible in 100 languages. *Language Resources and Evaluation*, *49*(2), 375–395.

Clark, S. (2002). Supertagging for Combinatory Categorial Grammar. *Proceedings of the 6th International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+ 6)*, 19–24. https://aclanthology.org/W02-2203.pdf

Clark, S., & Curran, J. R. (2003). Log-linear models for wide-coverage CCG parsing. *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, 97–104. https://aclanthology.org/W03-1013.pdf

Clark, S., & Curran, J. R. (2004a). The importance of supertagging for wide-coverage CCG parsing. *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, 282–288. https://aclanthology.org/C04-1041

Clark, S., & Curran, J. R. (2004b). Parsing the WSJ using CCG and log-linear models. *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 103–110. https://aclanthology.org/P04-1014.pdf

Clark, S., & Curran, J. R. (2007). Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, *33*(4), 493–552. https://doi.org/10.1162/coli.2007.33.4.493

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukçuoğlu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, *12*, 2493–2537.

Curran, J., Clark, S., & Bos, J. (2007). Linguistically motivated large-scale NLP with C&C and Boxer. *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, 33–36. https://aclanthology.org/P07-2009

Curry, H. B., Feys, R., Craig, W., Hindley, J. R., & Seldin, J. P. (1958). *Combinatory Logic* (Vol. 1). North-Holland Amsterdam.

Dahl, D. A., Bates, M., Brown, M., Fisher, W., Hunicke-Smith, K., Pallett, D., Pao, C., Rudnicky, A., & Shriberg, E. (1994). Expanding the scope of the atis task: The atis-3 corpus. *Proceedings of the Workshop on Human Language Technology*, 43–48. https://doi.org/10.3115/1075812.1075823

Dalrymple, M. (2001). *Lexical Functional Grammar*. Brill.

Davidson, D. (1967). The logical form of action sentences. In N. Rescher (Ed.), *The logic of decision and action*. University of Pittsburgh Press.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. https://doi.org/10.18653/v1/N19-1423

Eisner, J. (1996). Efficient normal-form parsing for Combinatory Categorial Grammar. *34th Annual Meeting of the Association for Computational Linguistics*, 79–86. https://doi.org/10.3115/981863.981874

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211.

Evang, K. (2019). Transition-based DRS parsing using stack-LSTMs. *Proceedings of the IWCS Shared Task on Semantic Parsing*. https://doi.org/10.18653/v1/W19-1202

Evang, K., Basile, V., Chrupała, G., & Bos, J. (2013). Elephant: Sequence labeling for word and sentence segmentation. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1422–1426. https://aclanthology.org/D13-1146.pdf

Fellbaum, C. (2010). WordNet. *Theory and applications of ontology: Computer applications* (pp. 231–243). Springer.

Frazier, L. (1979). *On comprehending sentences: Syntactic parsing strategies.* University of Connecticut.

Fu, Q., Zhang, Y., Liu, J., & Zhang, M. (2020). DRTS parsing with structure-aware encoding and decoding. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 6818–6828. https://doi.org/10.18653/v1/2020.acl-main.609

Geach, P. T. (1962). *Reference and Generality*. Cornell University Press London.

Geck, C. (2017). The World Factbook. *The Charleston Advisor*, *19*(1), 58–60.

Geurts, B., Beaver, D. I., & Maier, E. (2020). Discourse Representation Theory. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Spring 2020). Metaphysics Research Lab, Stanford University.

Geurts, B., & Maier, E. (2013). Layered Discourse Representation Theory. *Perspectives on linguistic pragmatics* (pp. 311–327). Springer.

Giampiccolo, D., Magnini, B., Dagan, I., & Dolan, B. (2007). The third PASCAL recognizing textual entailment challenge. *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 1–9. https://aclanthology.org/W07-1401

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., & Mikolov, T. (2018). Learning word vectors for 157 languages. *Proceedings of the 11th International Conference on Language Resources and Evaluation*. https://aclanthology.org/L18-1550

Heil, A. L. (2003). *Voice of america: A history*. Columbia University Press.

Herzig, J., Nowak, P. K., Müller, T., Piccinno, F., & Eisenschlos, J. (2020). TaPas: Weakly supervised table parsing via pre-training. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4320–4333. https://doi.org/10.18653/v1/2020.acl-main.398

Hobbs, J. R. (1985). *On the Coherence and Structure of Discourse* (tech. rep.). Center for the Study of Language and Information. Stanford University.

Hobbs, J. R., Stickel, M. E., Appelt, D. E., & Martin, P. (1993). Interpretation as abduction. *Artificial Intelligence*, *63*(1-2), 69–142.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hockenmaier, J. (2003). *Data and models for statistical parsing with Combinatory Categorial Grammar* (Doctoral dissertation). University of Edinburgh.

Hockenmaier, J., & Bisk, Y. (2010). Normal-form parsing for Combinatory Categorial Crammars with generalized composition and type-raising. *Proceedings of the 23rd International Conference on Computational Linguistics*, 465–473. https://aclanthology.org/C10-1053.pdf

Hockenmaier, J., & Steedman, M. (2007). CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, *33*(3), 355–396. https://doi.org/10.1162/coli.2007.33.3.355

Honnibal, M., Nothman, J., & Curran, J. R. (2009). Evaluating a statistical CCG parser on Wikipedia. *Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources (People's Web)*, 38–41. https://aclanthology.org/W09-3306

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 328–339. https://aclanthology.org/P18-1031.pdf

Huybregts, R. (1976). Overlapping dependencies in Dutch. *Utrecht Working Papers in Linguistics*, *1*, 24–65.

Ide, N., Baker, C., Fellbaum, C., & Passonneau, R. (2010). The manually annotated sub-corpus: A community resource for and by the people. *Proceedings of the ACL 2010 Conference Short Papers*, 68–73. https://aclanthology.org/P10-2013

Ide, N., & Suderman, K. (2004). The American National Corpus First Release. *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*. http://www.lrec-conf.org/proceedings/lrec2004/pdf/518.pdf

Iyer, S., Konstas, I., Cheung, A., Krishnamurthy, J., & Zettlemoyer, L. (2017). Learning a neural semantic parser from user feedback. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 963–973. https://doi.org/10.18653/v1/P17-1089

Jacobson, P. (1992). The lexical entailment theory of control and the tough construction. *Lexical Matters*, 269–299.

Joshi, A. K., & Schabes, Y. (1997). Tree-Adjoining Grammars. *Handbook of formal languages* (pp. 69–123). Springer.

Kallmeyer, L. (2010). *Parsing Beyond Context-Free Grammars*. Springer Science & Business Media.

Kamp, H. (1981). A theory of truth and semantic representation. In J. A. G. Groenendijk, T. M. V. Janssen, & M. B. J. Stokhof (Eds.), *Formal methods in the study of language* (pp. 277–322). Mathematisch Centrum.

Kamp, H., & Reyle, U. (1993). *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer, Dordrecht.

Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *35*(3), 400–401.

Kayadelen, T., Öztürel, A., & Bohnet, B. (2020). A gold standard dependency treebank for Turkish. *Proceedings of the 12th Language Resources and Evaluation Conference*, 5156–5163.

Kazimierz, A. (1935). Die syntaktische konnexitat. *Polish Logic*.

Kiss, T., & Alexiadou, A. (2015). *Syntax - Theory and Analysis* (Vol. 2). Walter de Gruyter GmbH & Co KG.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., & Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, 177–180. https://aclanthology.org/P07-2045

Kwiatkowksi, T., Zettlemoyer, L., Goldwater, S., & Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. *Proceedings of the 2010 Conference on Empirical Methods in Natural Lan-*

guage Processing, 1223–1233. https://www.aclweb.org/anthology/D10-1119.pdf

Kwiatkowski, T., Choi, E., Artzi, Y., & Zettlemoyer, L. (2013). Scaling semantic parsers with on-the-fly ontology matching. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1545–1556. https://aclanthology.org/D13-1161

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., & Steedman, M. (2011). Lexical generalization in CCG grammar induction for semantic parsing. *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 1512–1523. https://aclanthology.org/D11-1140

Lambek, J. (1958). The mathematics of sentence structure. *The American Mathematical Monthly*, *65*(3), 154–170.

Lascarides, A., & Asher, N. (2008). Segmented Discourse Representation Theory: Dynamic semantics with discourse structure. *Computing meaning* (pp. 87–124). Springer.

Le, P., & Zuidema, W. (2012). Learning compositional semantics for open domain semantic parsing. *Proceedings of the 24th International Conference on Computational Linguistics*, 1535–1552. https://aclanthology.org/C12-1094.pdf

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lewis, M., Lee, K., & Zettlemoyer, L. (2016). LSTM CCG parsing. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 221–231. https://doi.org/10.18653/v1/N16-1026

Lewis, M., & Steedman, M. (2014a). A* CCG parsing with a supertag-factored model. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 990–1000. https://doi.org/10.3115/v1/D14-1107

Lewis, M., & Steedman, M. (2014b). Improved CCG parsing with semi-supervised Supertagging. *Transactions of the Association for Computational Linguistics*, *2*, 327–338. https://doi.org/10.1162/tacl_a_00186

Liu, J., Cohen, S. B., & Lapata, M. (2018). Discourse Representation Structure parsing. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 429–439. https://doi.org/10.18653/v1/P18-1040

Liu, J., Cohen, S. B., & Lapata, M. (2019a). Discourse representation parsing for sentences and documents. *Proceedings of the 57th Annual Meeting of the*

Association for Computational Linguistics, 6248–6262. https://doi.org/10.18653/v1/P19-1629

Liu, J., Cohen, S. B., & Lapata, M. (2019b). Discourse Representation Structure parsing with recurrent neural networks and the Transformer model. *Proceedings of the IWCS Shared Task on Semantic Parsing*. https://doi.org/10.18653/v1/W19-1203

Liu, J., Cohen, S. B., & Lapata, M. (2020). Dscorer: A fast evaluation metric for Discourse Representation Structure parsing. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4547–4554. https://doi.org/10.18653/v1/2020.acl-main.416

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTta: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692. Version 1.*

Mann, W. C., & Thompson, S. A. (1987). *Rhetorical Structure Theory: A theory of text organization*. University of Southern California, Information Sciences Institute.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55–60. https://doi.org/10.3115/v1/P14-5010

Marcus, M. P., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, *19*(2), 313–330. https://aclanthology.org/J93-2004

Minnen, G., Carroll, J., & Pearce, D. (2001). Applied morphological processing of English. *Natural Language Engineering*, *7*(3), 207–223.

Miyao, Y., & Tsujii, J. (2002). Maximum entropy estimation for feature forests. *Proceedings of the 2nd International Conference on Human Language Technology Research*, 292–297.

Montague, R., & Thomason, R. H. (1975). Formal Philosophy: Selected Papers of Richard Montague. *Erkenntnis*, *9*(2).

Muller, P., Afantenos, S., Denis, P., & Asher, N. (2012). Constrained decoding for text-level discourse parsing. *Proceedings of the 24th International Conference on Computational Linguistics*, 1883–1900. https://aclanthology.org/C12-1115

Muskens, R. (1996). Combining Montague semantics and discourse representation. *Linguistics and Philosophy*, *19*, 143–186.

Nicosia, M., Qu, Z., & Altun, Y. (2021). Translate & Fill: Improving zero-shot multilingual semantic parsing with synthetic data. *Findings of the Association for*

Computational Linguistics: EMNLP 2021, 3272–3284. https://doi.org/10.18653/v1/2021.findings-emnlp.279

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajič, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., & Zeman, D. (2016). Universal Dependencies v1: A multilingual treebank collection. *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC'16)*, 1659–1666. https://aclanthology.org/L16-1262

Nivre, J., de Marneffe, M.-C., Ginter, F., Hajic, J., Manning, C. D., Pyysalo, S., Schuster, S., Tyers, F., & Zeman, D. (2020). Universal Dependencies v2: An evergrowing multilingual treebank collection. *Proceedings of the 12th International Conference on Language Resources and Evaluation (LREC)*, 4034–4043. https://aclanthology.org/2020.lrec-1.497.pdf

Parsons, T. (1980). Modifiers and quantifiers in natural language. *Canadian Journal of Philosophy Supplementary Volume*, *6*, 29–60.

Parsons, T. (1990). *Events in the Semantics of English: A Study in Subatomic Semantics*. MIT Press.

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1532–1543. https://doi.org/10.3115/v1/D14-1162

Pentus, M. (1993). Lambek grammars are context free. *Proceedings of 8th Annual IEEE Symposium on Logic in Computer Science*, 429–433.

Pentus, M. (2006). Lambek calculus is NP-complete. *Theoretical Computer Science*, *357*(1-3), 186–201.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2227–2237. https://doi.org/10.18653/v1/N18-1202

Polanyi, L. (1985). A theory of discourse structure and discourse coherence. *Papers from the General Session at the 21st Regional Meeting of the Chicago Linguistics Society, 1985*.

Pollard, C., & Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Prange, J., Schneider, N., & Srikumar, V. (2021). Supertagging the long tail with tree-structured decoding of complex categories. *Transactions of the Association for Computational Linguistics*, *9*, 243–260. https://doi.org/10.1162/tacl_a_00364

Press, O., & Wolf, L. (2017). Using the output embedding to improve language models. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 157–163. https://www.aclweb.org/anthology/E17-2025

Pyysalo, S., Ginter, F., Heimonen, J., Björne, J., Boberg, J., Järvinen, J., & Salakoski, T. (2007). Bioinfer: A corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, *8*(1), 1–24.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text Transformer. *J. Mach. Learn. Res.*, *21*(140), 1–67.

Ratnaparkhi, A. (1998). *Maximum entropy models for natural language ambiguity resolution*. University of Pennsylvania.

Reddy, S., Lapata, M., & Steedman, M. (2014). Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, *2*, 377–392. https://doi.org/10.1162/tacl_a_00190

Reddy, S., Täckström, O., Collins, M., Kwiatkowski, T., Das, D., Steedman, M., & Lapata, M. (2016). Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, *4*, 127–140. https://doi.org/10.1162/tacl_a_00088

Reinhart, T. (1976). *The Syntactic Domain of Anaphora* (Doctoral dissertation). Massachusetts Institute of Technology.

Reinhart, T. (1981). Definite NP anaphora and c-command domains. *Linguistic Inquiry*, *12*(4), 605–635.

Reinhart, T. (1983). *Anaphora and Semantic Interpretation*. Crook Helm London.

Rimell, L., & Clark, S. (2008). Adapting a lexicalized-grammar parser to contrasting domains. *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 475–484. https://aclanthology.org/D08-1050.pdf

Sayeed, A., & Demberg, V. (2012). Incremental neo-Davidsonian semantic construction for TAG. *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, 64–72. https://aclanthology.org/W12-4608

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80.

Schuler, K. K. (2005). *VerbNet: A broad-coverage, comprehensive verb lexicon*. University of Pennsylvania.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, *45*(11), 2673–2681.

Sekine, S., Sudo, K., & Nobata, C. (2002). Extended named entity hierarchy. *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC'02)*. http://www.lrec-conf.org/proceedings/lrec2002/pdf/120.pdf

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1715–1725. https://doi.org/10.18653/v1/P16-1162

Shieber, S. M. (1985). Evidence against the context-freeness of natural language. *Philosophy, language, and artificial intelligence* (pp. 79–89). Springer.

Sipser, M. (1996). *Introduction to the Theory of Computation*. ACM New York, NY, USA.

Steedman, M. (1993). Categorial Grammar. *Lingua*, *90*(3), 221–258.

Steedman, M. (1996). Surface Structure and Interpretation. *Number 30 in Linguistic Inquiry Monographs*. MIT Press.

Steedman, M. (2000). *The Syntactic Process*. MIT press Cambridge, MA.

Steedman, M., & Baldridge, J. (2011). Combinatory Categorial Grammar. *Non-transformational syntax: Formal and explicit models of grammar* (pp. 181–224). Wiley Online Library. https://doi.org/10.1002/9781444395037.ch5

Tang, L. R., & Mooney, R. J. (2001). Using multiple clause constructors in inductive logic programming for semantic parsing. *European Conference on Machine Learning*, 466–477.

Tenney, I., Das, D., & Pavlick, E. (2019). BERT rediscovers the classical NLP pipeline. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4593–4601. https://doi.org/10.18653/v1/P19-1452

Tian, Y., Song, Y., & Xia, F. (2020). Supertagging Combinatory Categorial Grammar with attentive graph convolutional networks. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6037–6044. https://doi.org/10.18653/v1/2020.emnlp-main.487

Turian, J., Ratinov, L.-A., & Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 384–394. https://aclanthology.org/P10-1040

Van der Sandt, R. A. (1992). Presupposition projection as anaphora resolution. *Journal of Semantics*, *9*(4), 333–377.

van Noord, R., Abzianidze, L., Haagsma, H., & Bos, J. (2018). Evaluating scoped meaning representations. *Proceedings of the 11th International Conference on Language Resources and Evaluation*. https://www.aclweb.org/anthology/L18-1267

van Noord, R., Abzianidze, L., Toral, A., & Bos, J. (2018). Exploring neural methods for parsing discourse representation structures. *Transactions of the Association for Computational Linguistics*, *6*, 619–633. https://doi.org/10.1162/tacl_a_00241

van Noord, R., Toral, A., & Bos, J. (2019). Linguistic information in neural semantic parsing with multiple encoders. *Proceedings of the 13th International Conference on Computational Semantics - Short Papers*, 24–31. https://doi.org/10.18653/v1/W19-0504

van Noord, R., Toral, A., & Bos, J. (2020). Character-level representations improve DRS-based semantic parsing even in the age of BERT. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 4587–4603. https://doi.org/10.18653/v1/2020.emnlp-main.371

Vaswani, A., Bisk, Y., Sagae, K., & Musa, R. (2016). Supertagging with LSTMs. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 232–237. https://aclanthology.org/N16-1027.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). Graph Attention Networks. *arXiv preprint arXiv:1710.10903. Version 3.*

Venhuizen, N. J., Basile, V., Evang, K., & Bos, J. (2013). Gamification for word sense labeling. *Proceedings of the 10th International Conference on Computational Semantics (Short Papers)*, 397–403. https://aclanthology.org/W13-0215

Venhuizen, N. J., Bos, J., & Brouwer, H. (2013). Parsimonious semantic representations with projection pointers. *Proceedings of the 10th International Conference on Computational Semantics (Long Papers)*, 252–263. https://aclanthology.org/W13-0122.pdf

Venhuizen, N. J., & Brouwer, H. (2014). PDRT-SANDBOX: An implementation of Projective Discourse Representation Theory. *Proceedings of the 18th Work-*

shop on the Semantics and Pragmatics of Dialogue (DialWatt-SemDial 2014), Edinburgh, 249–51.

Vijay-Shanker, K., & Weir, D. J. (1990). Polynomial time parsing of Combinatory Categorial Grammars. *28th Annual Meeting of the Association for Computational Linguistics*, 1–8. https://doi.org/10.3115/981823.981824

Vijay-Shanker, K., & Weir, D. (1993). Parsing some constrained grammar formalisms. *Computational Linguistics*, *19*(4), 591–636. https://aclanthology.org/J93-4002.pdf

Vijay-Shanker, K., & Weir, D. (1994). The equivalence of four extensions of Context-Free Grammars. *Mathematical Systems Theory*, *27*(6), 511–546.

Vijay-Shanker, K., Weir, D., & Joshi, A. (1987). Characterizing structural descriptions produced by various grammatical formalisms. *25th Annual Meeting of the Association for Computational Linguistics*, 104–111. https://aclanthology.org/P87-1015.pdf

Webber, B. L. (1988). *Discourse Deixis and Discourse Processing* (tech. rep.). Department of Computer and Information Science. University of Pennsylvania.

Webber, B. L. (2004). D-ltag: Extending lexicalized tag to discourse. *Cognitive Science*, *28*(5), 751–779.

Weir, D. (1988). *Characterizing Mildly Context-Sensitive Grammar Formalisms* (Doctoral dissertation). University of Pennsylvania.

XTAG Research Group. (1995). A Lexicalized Tree Adjoining Grammar for English. *University of Pennsylvania, IRCS Report 95-03*.

Xu, W. (2016). LSTM shift-reduce CCG parsing. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1754–1764. https://doi.org/10.18653/v1/D16-1181

Xu, W., Auli, M., & Clark, S. (2015). CCG supertagging with a recurrent neural network. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 250–255. https://doi.org/10.3115/v1/P15-2041

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., & Radev, D. (2018). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 3911–3921. https://doi.org/10.18653/v1/D18-1425

Zaenen, A., Carletta, J., Garretson, G., Bresnan, J., Koontz-Garboden, A., Nikitina, T., O'Connor, M. C., & Wasow, T. (2004). Animacy encoding in English: Why and how. *Proceedings of the Workshop on Discourse Annotation*, 118–125. https://aclanthology.org/W04-0216

Zelle, J. M., & Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, 1050–1055.

Zettlemoyer, L. (2009). *Learning to map sentences to logical form* (Doctoral dissertation). Massachusetts Institute of Technology.

Zettlemoyer, L., & Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, 658–666. https://dl.acm.org/doi/abs/10.5555/3020336.3020416

Zettlemoyer, L., & Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 678–687. https://www.aclweb.org/anthology/D07-1071.pdf

Zhang, Y., & Clark, S. (2011). Shift-reduce CCG parsing. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 683–692. https://aclanthology.org/P11-1069

Zhong, V., Xiong, C., & Socher, R. (2017). Seq2Sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103. Version 7.*

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Öztürel, İ. Adnan
E-mail: adnanozturel@gmail.com

EDUCATION

Degree	Institution	Year of Graduation
M.Sc.	Cognitive Science, METU	2011
B.Sc.	Computer Science, Bilkent University	2008
High School	TED Ankara Koleji	2004

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2014-Ongoing	Google Research	Computational Linguist
2008-2014	Information Systems, METU	Research Assistant

PUBLICATIONS

International Conference Publications

Kayadelen, T., Öztürel, A., & Bohnet, B. (2020). A gold standard dependency treebank for Turkish. *Proceedings of the 12th International Conference on Language Resources and Evaluation (LREC)* (pp. 5156-5163).

Öztürel, A., Kayadelen, T., & Demirşahin, I. (2019). A syntactically expressive morphological analyzer for Turkish. *Proceedings of the 14th International Conference on Finite-State Methods and Natural Language Processing (FSMNLP)* (pp. 65-75).

Andersson, M., Öztürel, A., & Pareti, S. (2016). Annotating topic development in information seeking queries. *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC)* (pp. 1755-1761).

Demirşahin, I., Öztürel, A., Bozşahin, C., & Zeyrek, D. (2013). Applicative structures and immediate discourse in the Turkish Discourse Bank. *Proceedings of the 7th*

Linguistic Annotation Workshop and Interoperability with Discourse (pp. 122-130).

Öztürel, A., & Bozşahin, C. (2012) Musical agreement via social dynamics can self-organize a closed community of music: A computational model. *Proceedings of the 12th International Conference on Music Cognition and the 8th Triennial Conference of the European Society for Cognitive Sciences of Music* (pp. 766-775).

Öztürel, A. & Özcan, O. O. (2012). Collective learning of an emergent vocabulary: Naming game with reinforcement learning. *First Central European Conference in Linguistics for Postgraduate Students* (Vol. 225, p. 225).

Thesis

Öztürel, A. (2011). *A computational model of social dynamics of musical agreement*. (MSc Dissertation, Middle East Technical University, Ankara, Turkey).