A FUZZY PETRI NET MODEL FOR INTELLIGENT DATABASES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BURÇİN BOSTAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR

THE DEGREE OF DOCTOR OF PHILOSOPHY

IN

COMPUTER ENGINEERING

MARCH 2005

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Ayşe Kiper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Adnan Yazıcı
Supervisor

Examining Committee Members

Prof. Dr. Özgür Ulusoy	(Bilkent University)	
Prof. Dr. Adnan Yazıcı	(METU, CENG)	
Assoc. Prof. Dr. İsmail Hakkı Toroslu	(METU, CENG)	
Assoc. Prof. Dr. Nihan Kesim Çiçekli	(METU, CENG)	
Assist. Prof. Dr. Halit Oğuztüzün	(METU, CENG)	

Name, Lastname	: Burçin Bostan
Signature	:

ABSTRACT

A FUZZY PETRI NET MODEL FOR INTELLIGENT DATABASES

Bostan, Burçin

Ph.D., Department of Computer Engineering

Supervisor: Prof. Dr. Adnan Yazıcı

March 2005, 181 pages

Knowledge intensive applications require an intelligent environment, which can perform deductions in response to user queries or events that occur inside or outside of the applications. For that, we propose a Fuzzy Petri Net (FPN) model to represent the knowledge and the behavior in an intelligent object-oriented database environment, which integrates fuzzy, active and deductive rules with database objects. By gaining intelligent behaviour, the system maintains objects to perceive dynamic occurences and user queries. Thus, objects can produce new knowledge or keep themselves in a consistent, stable, and upto-date state.

The behavior of a system can be unpredictable due to the rules triggering or untriggering each other (non-termination). Intermediate and final database states may also differ according to the order of rule executions (non-confluence). In order to foresee and solve problematic behavior patterns, we employ static rule analysis on the FPN structure that provides easy checking of the termination property without requiring any extra construct. In addition, with our proposed inference algorithm, we guarantee confluent rule executions.

The techniques and solutions provided in this study can be utilized in various complex systems, such as weather forecasting applications, environmental information systems, defense applications, video database applications, etc. We implement a prototype of the model for the weather forecasting of the Central Anatolia Region.

Keywords: object-oriented database, knowledge-base, fuzziness, active database, fuzzy Petri net, static rule analysis, termination, confluence.

ÖZ

AKILLI VERİTABANLARI İÇİN BULANIK BİR PETRİ NET MODELİ

Bostan, Burçin
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Adnan Yazıcı

Mart 2005, 181 sayfa

Bilgi yoğun uygulamalar kullanıcı sorgulamalarından ya da uygulamanın içinde ya da dışında gerçekleşen olaylardan çıkarımlar yapabilen akıllı ortamlara ihtiyaç duyarlar. Bu nedenle, bilgiyi göstermek ve davranışı modellemek için bulanık, aktif, çıkarımsal kuralların veritabanı nesneleri ile birleştiği nesneye yönelik akıllı bir veritabanı ortamında Bulanık Petri Net modeli öneriyoruz. Bu modelle akıllı davranış kazanan sistem, nesnelerin dinamik olayları ve kullanıcı sorgulamalarını algılayabilmesini sağlar. Böylece nesneler yeni bilgi üretebilir, ve kendilerini tutarlı, dayanıklı, güncel tutabilirler.

Bir sistemin davranışı kuralların birbirini tetiklemesi ya da engellemesi (sonlanamama) nedeniyle tahmin edilemez olabilir. Ara ve nihai veritabanı durumları da kuralların işlenme sırasına göre değişiyor olabilir (tutarsızlık). Bu problemli davranış şekillerini önceden görerek çözümleyebilmek için Bulanık Petri Net'ler üzerinde sonlanma özelliğinin ek yapı ihtiyacı doğurmadan kolayca kontrolünü sağlayan statik kural analizi çalıştırıyoruz. Ayrıca, önerdiğimiz çıkarım algoritması ile tutarlı kural işlenişini garanti ediyoruz.

Bu çalışmada sağlanan teknikler ve çözüm yolları hava tahmini uygulamaları, çevre bilgi sistemleri, savunma uygulamaları, görüntü veritabanı uygulamaları gibi çeşitli kompleks uygulamaların modellenmesinde kullanılabilir. Modelin prototipi İç Anadolu Bölgesinin hava tahmini için gerçekleştirilmiştir.

Anahtar Kelimeler: nesneye-yönelik veritabanı, bilgi-tabanı, bulanıklık, aktif veritabanı, bulanık Petri net, statik kural analizi, sonlanma, tutarlılık

To my parents, Berrin - Ahmet Bostan

and

my lovely grandmother, Afife Tümer

ACKNOWLEDGMENTS

First of all, I am honored to present my special thanks to my supervisor, Prof. Dr. Adnan Yazıcı for his encouragement and guidance throughout this study. His continuous support and insight made this thesis possible.

I would like to thank to my thesis jury members Prof. Dr. Özgür Ulusoy, Assoc. Prof. Dr. Nihan Kesim Çiçekli, Assist Prof. Dr. Halit Oğuztüzün and Assoc. Prof. Dr. İsmail Hakkı Toroslu for their valuable comments and guidance.

I would like to thank to Mehmet Yayvan, weather forecast expert from the Turkish State Meteorological Service, who helped me to learn the weather events and extract the commonsense rules of Weather Forecast.

I would like to thank to my colleagues in Central Bank of Turkey. Being in the same environment and same situation, Arzu Şişman Yılmaz, Aziz Sözer and Cemil Ulu, we all motivated each other for the success of our Phd. studies. Thanks to them for those encouraging and reassuring chats. I also would like to thank to Selma Erbay, Turgut Tatlıdil and İlker Durusoy for their patience and continuous motivation.

At last, but not the least, I would like to thank to my family for their patience, love and belief in me. I am especially grateful to my beloved husband İbrahim and brother Yalçın, who provided everything I needed. Without their patience, continuous assistance and long-time friendship, I would not have had the strength to complete this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

PN	Petri Net
FPN	Fuzzy Petri Net
CPN	Colored Petri Net
TG	Triggering Graph
AG	Activation Graph

CHAPTER 1

INTRODUCTION

1.1 Scope of the Study

Up to now, the object oriented database model has been very successful in modelling complex applications [4, 32, 43]. However, powerful database modeling features of object oriented databases do not suffice for modeling the complexities of knowledge intensive applications such as spatio-temporal databases, multimedia databases, workflow applications, defense applications, which require an intelligent environment that can perform deductions [49]. In such an environment, there can be two reasons for deduction; one of them is user queries and the other is events occurring inside or outside of the system.

In an object-oriented database, incorporation of knowledge for perceiving state changes and answering queries with deductive capabilities can be handled via two different ways. One of them is polling the system periodically for interesting occurrences of events [17, 10]. However, the problem with the polling approach is that either the database will be overloaded with queries that will fail most of the time, or dynamic occurrences of events may be missed. The other approach is embedding some intelligent code into method bodies [6, 45]. However, this solution leads the applications to lose their modularity, which makes them harder to validate and difficult to maintain. Due to the weaknesses of both approaches, active object oriented databases are developed as a solution to perceive dynamic behavior by including event-driven facilities that are necessary for implementing system reactions by integrating rules within a database [17, 35, 37, 63].

Most of the existing database models, including the active object-oriented ones are designed under the assumptions of precision. However, knowledge intensive applications often involve complex information with uncertainty. In general, data and information kept in databases may be uncertain for the following reasons: (1) If data is obtained from different sources, due to possible semantic differences it may not be feasible to represent such data in a precise way. Otherwise, either true data is lost, or false and useless information is obtained. (2) Decision making in many knowledge-intensive applications usually involves various forms of uncertainty. (3) Information sometimes represents subjective opinions and judgments, which is inherently both complex and uncertain. (4) When conveying vague information in natural languages, numerous linguistic terms with modifiers (e.g. "very", "more or less", etc.) and quantifiers (e.g. "many", "few", "most", etc.) are used.

In our research effort, different parts of which are included in [11, 12, 13], we propose a Fuzzy Petri Net model for an intelligent object-oriented database environment [49, 12] in order to fulfill the requirements of knowledge-intensive applications. For this, we integrate fuzzy, active and deductive rules together with an inference mechanism in a fuzzy object-oriented database environment. Our architecture fulfills the requirements of complex real world applications such as weather forecasting, environmental monitoring, defense applications, multimedia database applications [38, 2]. In a weather forecasting application there exists huge amounts of data related to the atmospheric elements, such as *pressure*, *temperature*, *humidity*, coming from sensors connected to weather stations. Since a couple of changes taking place at the same time are the indications of some forthcoming events, the system needs to perform deductions in order to determine the possible results of these changes in the environment. For example, *high temperature*, *low pressure* and *high humidity* in certain location and time may trigger *heavy rain*, which can be represented as a fuzzy rule in our system. A particular value change on an atmospheric element often triggers multiple fuzzy rules, which should be executed concurrently to generate result representing the deduction of all the relevant rules, which require a fuzzy inference mechanism. In addition there could be state changes in the application environment, such as seasonal changes. In winter, we expect weather events like *snow*, *freeze* etc., while in spring we expect *rain*, *hail* or *shower*. The application should be intelligent enough to handle these state changes, i.e., give more importance to some rules or prune some others according to the state. There may also be user queries which require deductions. These queries may be on the current values of the elements of the application domain. For example, in a weather forecasting application, we can have a query like *"Which cities are in emergency state with respect to the heavy snow?"*. We may also want to know the future trends of the elements of the application domain. For example, we may ask queries like *"Which cities are expected to be affected very highly from the coming storm?"* or *"What is the expected weather event in Paris these days?"*. We can increase such examples for many other real world applications. In order to satisfy the requirements of such knowledge intensive applications, we integrate fuzzy, active and deductive rules with database objects. That is, our intelligent database environment allows objects to perceive dynamic occurrences or user queries after which they produce new knowledge or keep themselves in a consistent, stable, and up-to-date state; thus, performing intelligent behavior.

In literature, it has been argued that integration of active and deductive paradigms into a unique homogeneous framework is an important and challeging goal [81]. There have been a number of studies dealing with the problem of defining a unified semantics for deductive and active rules. In that, there exist two research directions for integration: some of them [81, 9] try to extend deductive databases to support active behavior, while others [20] study how deductive rules can be implemented by means of active rules. For example, Zanilo [81] uses a non-monotonic extention of logical clauses, which includes negation and aggregates under XY-stratification semantics. In that, active rules are expressed by means of built-in predicates which implement basic update operations. Bayer et al. [9] specify a framework for supporting triggers in the context of deductive databases. While event specification is defined on insertion or deletion of facts to predicates and on their composition, condition specification uses extended datalog with negation and built-in predicates. A different line of research is presented by Ceri et al. [20], which uses production rules to physically maintain intentional data defined by the deductive rules. In that study, active rules are derived automatically in order to maintain intentional data when extensional relations are updated by users. In our study, users can express active and deductive rules independently in their traditional form. However, we consider deductive rules as special cases of active rules, where we use abstract kind of events. Therefore, internally all rules are of active kind and their processing is modelled with a Fuzzy Petri net.

In our study, we use Fuzzy Petri Nets (FPNs) to represent knowledge and model the behavior of the system. Also, we check the properties of the system, i.e., perform static rule analysis, using our FPN. Petri Nets, in general, are considered as a graphical and mathematical modeling tool. They are powerful in describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel and nondeterministic [54]. Several kinds of Petri Nets, such as FPNs, CPNs [44], have been investigated as tools for representing rules in knowledge-based systems. The main advantage of using Petri nets in rule-based systems is that they provide a structured knowledge representation in which the relationships among the rules in the knowledge base are easily understood, and they render a systematic inference capability [24].

Using PNs to model rule based reasoning provides a couple of advantages:

- PNs in general are good at describing and analyzing the flow of information and control in systems, particularly systems that may exhibit sequentiality, concurrency or synchronization of events [65];

- PNs' graphical representation, using a few types of elements, can help to understand the model, and construct and modify rule bases;

- Since PNs can be used to represent events in a top-down fashion, they can be used to model a composite event hierarchically from simpler event models;

- They can model the dynamic behavior of rule-based reasoning. Evaluation of markings is used to simulate the dynamic behavior of the system. The explanation of how to reach conclusions is expressed through the movements of tokens in PNs. At any time during the interpretation process, position of tokens in the PN summarize what happened in the past and predict what will happen in the future providing an incremental composition of events;

- PNs' analytic capability can help with checking properties of a modeled system to gain deeper insights into the system.

In order to represent uncertain and imprecise knowledge existing in various knowledge-intensive applications, the degree of truth of rules and facts represented in a knowledge base is expressed as a real number in interval [0,1]. Fuzzy Petri Nets (FPNs) are formed to handle such fuzzy expressions [24]. FPNs eliminate the need to scan all the rules. They improve the efficiency of fuzzy rule-based reasoning by using transitions and arcs to connect fuzzy rules as a net-based structure. There have been a couple of approaches on alternative formulations of FPNs to model the behavior of a system. However, due to the unstructured and unpredictable nature of rule processing, rules can be difficult to program and the behavior of a system can be complex and sometimes unpredictable. In an active database, rules may trigger and untrigger each other, and the intermediate and final states of the database can depend upon which rules are triggered and executed in which order. In order to determine these undesirable behavior patterns of the rule base, static rule analysis should be performed [1]. Such an analysis involves identifying certain properties of the rule base at compile-time, which gives the programmer an opportunity to modify the rule base.

Two important and desirable properties of active rule behavior are termination and confluence [1]. These properties are defined for user-defined changes and database states in a given rule set.

- *Termination:* A rule set is guaranteed to terminate if, for any database state and initial modification, rule processing does not continue forever (i.e. rules do not activate each other indefinitely).

- *Confluence:* A rule set is confluent if, for any database state and initial modification, the final database state after rule processing is unique, i.e., it is independent of the order in which activated rules are executed.

Static rule analysis techniques only give sufficient conditions for guaranteeing the property searched for. For example, the identification of potential non-termination in a set of rules indicates the possibility of infinite loops at run time, while the identification of potential non-confluence indicates that a rule base may exhibit nondeterministic behavior.

1.2 Power of Petri Net within Alternatives

The success of any model is due to two factors: its modeling power and its decision power [65]. Modeling power refers to the ability to correctly represent the system. Decision power refers to the ability to determine properties of the modeled system. These two factors generally work at opposite direction; for example, when we increase modeling power (also complexity of the models and the modeled systems), ability to algoritmically determine the properties of the models is generally decreased.

Two subclasses of Petri Nets generally considered are *finite state machines* and *marked graphs*. Finite state machines are Petri Nets which are restricted so that each transition has exactly one input and one output place. A marked graph, on the other hand, is a Petri Net in which each input place has exactly one input transition and one output transition. Both of them have high decision power, but less modeling power compared to Petri Nets. An attempt to use automata has been made in Ode [36] based on the observation that event expressions are equivalent to regular expressions if they are not parameterized. This means that, Finite Automata are not sufficient in case of event parameters have to be supported. They have to be extended with a data structure that remembers the event parameters of the primitive events from the time of their occurence to the time at which the composite event gets signalled.

1.3 Summary of Contributions

The main contributions of this thesis can be summarized as follows:

- Incorporation of fuzziness in defining data, attributes, objects, classes, rules and their inference in an active database model;
- Introducing a model for fuzzy inference in fuzzy active rules where a model for scenario concept is developed;
- Development of a Fuzzy Petri Net model for fuzzy rule-based reasoning where:
  - Event and condition compositions are modeled;
  - Functionalities of transitions are enhanced to make them capable of performing fuzzification, event/condition composition, concurrent execution and combination in addition to the sup-min composition;
  - Parameter passing is provided which provides values of conditions and actions to be calculated from the parameters of events;
  - An algorithm for fuzzy rule-based inference is provided.
- Performing static rule analysis on the constructed FPN to check the properties of the modeled system, where:
  - A termination analysis algorithm, which also considers event/condition compositions, is provided to determine the true cycles; and
  - Confluent rule executions are guaranteed with the provided fuzzy inference algorithm.
- Implementing the whole environment resulting in a prototype system;
- Applying our proposed environment into an application, weather forecasting, for proof of concept.

1.4 Organization of the Thesis

This thesis is organized as follows: Chapter 2 gives some background information, which includes fuzzy logic, object-oriented database model, active databases, Petri nets, static rule analysis, to form a basis for this thesis. This chapter also contains the summary of the previous work related to the concepts that we study together with the contribution of this study. Chapter 3 describes how we incorporate fuzziness into active rules. Our FPN model for fuzzy rule-based reasoning is presented in Chapter 4. Chapter 5, explains how we check the properties of the modeled system by using FPN. The description of the prototype implementation of our model is given in Chapter 6. Chapter 7 presents an example application, which is weather forecasting, chosen as the case study of the model developed in this thesis. Finally, Chapter 8 gives the conclusion and states the future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, an introduction to the basic concepts of this study is given. We explain some background information on fuzzy concepts in Section 2.1. We give an overview of object-oriented database model in Section 2.2. Basic aspects of active database systems is given in Section 2.3. We describe Petri Nets in Section 2.4. These are followed by a brief summary of the assumptions and understandings in Static Rule Analysis in Section 2.5. Finally, we give a summary of the previous work related to the concepts that we study together with the contribution of this study in Section 2.6.

2.1 Basic Concepts of Fuzzy Logic

Since information is usually inexact and imprecise, trying to reduce every sort of data into a precise form causes something to be missed. Also, in case of the lack of precise information it is often the case that some relevant information is available. In these cases, it may be advantageous to design methods for impreciseness by which this information can be stored, manipulated and retrieved.

Fuzziness is a term that expresses an ambiguity that can be found in the definition of a concept or the meaning of a word or phrase. The term *young* or *tall* can be qualified as fuzzy. The process of solving problems that deal with ambiguous data using multivalued logic to represent crisp logic system is called

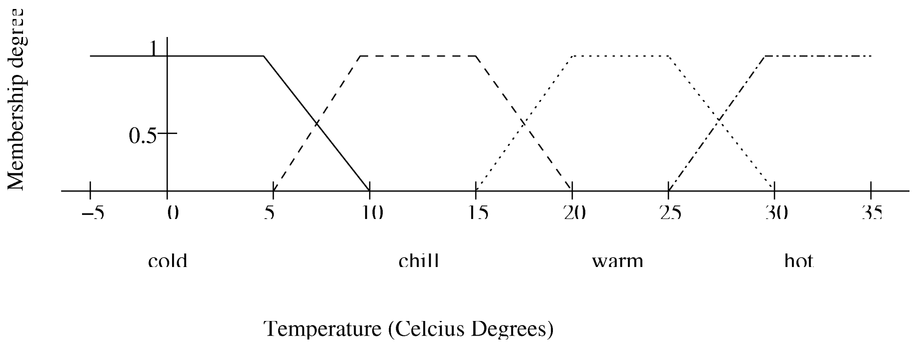


Figure 2.1: Membership Graph of Temperature Attribute

fuzzy logic. Fuzzy logic considers that all things are a matter of degree.

Fuzzy logic is based on four main concepts: *fuzzy sets*, *linguistic variables*, *possibility distribution* and *fuzzy if-then rules* [78].

Fuzzy Sets: The theory of Fuzzy sets was introduced by Zadeh [80] in order to handle imprecise, vague notions. Fuzzy set is a set with a smooth boundary, where classical set theory is generalized to allow partial membership. The degree of membership in such a set is expressed by a real number between 0 and 1; 0 means entirely not member of the set, 1 means completely member of the set, and a number in between means partially member of the set. *Membership function* maps objects in a domain of concern to their membership value. The membership function of a fuzzy set A is denoted as μ_A and the membership value of x in A is denoted as $\mu_A(x)$. The domain of membership function is called the *universe of discourse*.

Let U denotes the universe of discourse, and A is a fuzzy set on U. Membership function μ_A is defined as follows:

$$\mu_A(x) : U \longrightarrow [0, 1] \tag{2.1}$$

All elements of A belong more or less to A with a degree $\mu_A(x)$, where $\mu_A(x) > 0$.

Assume that we have a universe of discourse U for the temperature attribute values. Membership functions of fuzzy sets *cold*, *chill*, *warm* and *hot* are shown graphically in Figure 2.1.

Within them, membership function of *hot*, $\mu_{hot}(x)$, can be defined as:

$$\mu_{hot}(x) = \begin{cases} 0, & \text{if } x < 25 \\ (x-25)/5, & \text{if } 25 \leq x < 30 \\ 1, & \text{if } x \geq 30 \end{cases} \tag{2.2}$$

As shown in Figure 2.1, the fuzzy sets defined on a universe of discourse U may overlap each other and cover all the U. There may not be sudden switching from one set to another. A *temperature* of 28^oC is a "*hot weather*" with a membership degree of 0.6. On the other hand, it is a "*warm weather*" with a membership degree of 0.4. Calculation of the membership functions of the union, intersection and difference of two fuzzy sets is explained in [78].

Linguistic Variables: Like a conventional set, a fuzzy set can be used to describe the value of a variable. For example, the sentence "*The height of Hasan is tall*" uses a fuzzy set *tall* to describe the height of Hasan. The variable *height* in this example demonstrates another important concept of fuzzy logic: the *linguistic variable*. A linguistic variable enables its value to be described both qualitatively by a linguistic term and quantitatively by a corresponding membership function.

Possibility Distributions: In a crisp set, assigning a linguistic variable constrains the value of a variable with a sharp boundary between possible versus impossible values. However, fuzzy logic generalizes the binary distinction between possible vs. impossible to a matter of degree called the *possibility*. For instance, if we assign the fuzzy set *young* to the age of a person, whose membership function is given in Figure 2.2, we obtain a distribution about the possibility degree of an age. Π denotes a possibility distribution of age and x is a variable representing a person's age. In general, when a fuzzy set A is assigned to a variable X, the assignment results in a possibility distribution of X, which is defined by A's membership function:

$$\Pi_X(x) = \mu_A(x) \tag{2.3}$$

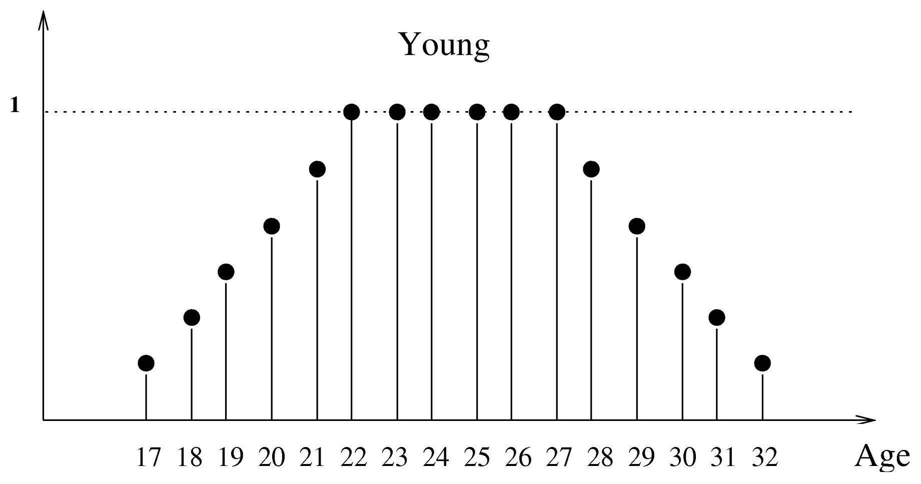


Figure 2.2: Possibility distribution of the fuzzy set *young*

Fuzzy Rules: A fuzzy rule is a knowledge representation schema for capturing knowledge that is imprecise and inexact by nature. A fuzzy rule has two components: an if-part (antecedent) and a then-part (consequent):

IF <antecedent> THEN <consequent>

The structure of a fuzzy rule is identical to that of a conventional rule. However, the antecedent of a fuzzy rule describes an elastic condition while the antecedent of a conventional rule describes a rigid condition.

Fuzzy rule-based inference is a generalization of *modus ponens*, where the inferred conclusion of modus ponens is modified by the degree to which the antecedent is satisfied. It consists of a set of fuzzy rules. Fuzzy rule-based inference has three major feature: First it enables a rule that partially matches the input data to make an inference. Second, it typically infers the possibility distribution of an output variable from that of input variable. Third, there exists a set of fuzzy rules with partially overlapping conditions. Therefore, a particular input to the system often triggers multiple fuzzy rules, inferred conclusions of which are combined to form an overall conclusion. The steps of fuzzy rule-based inference is given below:

1. *Fuzzy Matching*: Calculates the degree to which the input data matches the condition of the fuzzy rules.

2. *Inference*: Calculates the rule's conclusion based on its matching degree.

3. *Combination*: Combines the conclusions inferred by all fuzzy rules into a final conclusion.

4. *Defuzzification* (optional): Converts a fuzzy conclusion into a crisp one. It is generally performed in control systems.

2.1.1 Fuzzy Logic in Database Systems

Fuzzy logic has been used to extend database systems to capture imprecise information. This provided storing, manipulating and querying imprecise data [47, 59, 60, 78]. Imprecise information in a database system can be classified into two types: (1) imprecise attribute values in a tuple, and (2) partial membership of a tuple in a relation.

2.1.1.1 Approaches for Representing Imprecise Attribute Values

There are two approaches for representing fuzzy data in database systems as similarity-based approach and possibility-based approach.

Similarity-based Fuzzy Relations:
In this approach, similarity matrix is used to show the relationships of the fuzzy linguistic terms within a fuzzy domain with one another. A fuzzy relation S on domain X is a similarity relation for x, y, $z \in X$ if the following conditions hold:

1. Reflexivity: $\mu_S(x, x) = 1$

2. Symmetry: $\mu_S(x, y) = \mu_S(y, x)$

3. Transivity: $max_{y \in X}(min(\mu_S(x, y), \mu_S(y, z))) \leq \mu_S(x, z)$

An example similarity relation for age attribute is shown in Table 2.1. It satisfies the properties stated above.

If a query is executed to find "*people who are young with a similarity of 0.6*", the ones which are child or young are returned as an answer.

Table 2.1: Similarity Matrix for Age Attribute

Age	child	young	middle-aged	old
child	1	0.7	0.2	0.2
young	0.7	1	0.2	0.2
middle-aged	0.2	0.2	1	0.9
old	0.2	0.2	0.9	1

Possibility-based Fuzzy Relations:

The possibility-based approach is based on the possibility distribution theory. A possibility-based fuzzy relation generalizes a relation by allowing the value of an attribute A to be a possibility distribution $\Pi_A(t)$ of the attribute's domain.

Suppose that in a crime, the possibility distribution of the age of the suspect is:

$\Pi_{age}(suspect) = 0.5/20 + 1/25 + 0.8/30$

Suppose also that the membership function of the fuzzy set *young* is defined as a discrete fuzzy set as follows:

$young = 0.1/15 + 1/20 + 0.8/25 + 0.4/30$

Then the possibility degree for the suspect to satisfy the condition young is calculated as:

$$\begin{aligned} Possibility(young|\Pi_{age}(suspect)) &= Max\ (Min(0.5, 1), Min(1, 0.8), Min(0.8, 0.4) \\ &= Max(0.5, 0.8, 0.4) = 0.8 \end{aligned}$$

2.1.1.2 Partial Membership in a Relation

The second type of imprecise information in a database is the partial membership of a tuple in a relation. Membership in a relation is either 0 or 1 in a conventional database. However, there are relations whose membership has a gray area. For example, the relation *endangered_species* may include wild animals whose quantity has been significantly decreasing over the years but have not been officially declared as endangered species. These animals can be considered as somewhat endangered, hence a partial membership of the *endangered_species* relation.

2.2 Object-oriented Database Model

All entities and concepts in the real world are modeled as objects. Conceptually a unique *object identifier (oid)* is assigned to each object and this association between oid and class remains fixed even if attributes of the object change in time. The use of objects and oids permit OODBSs to share information gracefully; an object *o* can be shared by many objects simply by referencing the oid of *o*. So objects are nested through the referencing mechanism. An object has intrinsic characterizing properties called *attributes*. An object stores data in attributes in order to save its state, and responds to instructions for carrying out specific operations on its attributes when a request for that operation, *message*, is received by the object.

All objects which have the same attributes and methods are grouped into a higher level object, called a *class*. An object is called an *instance* of its class. An object created in one class belongs to all of its superclasses.

OODB models permit the organization of classes into a hierarchy called *isa* relationship. The objective of organizing objects in a hierarchy of classes is to share properties of objects in useful, economical and meaningful ways. Properties of a superclass can be inherited by its subclasses. However, selective inheritance occurs when a subclass inherits only some of the properties of a superclass. Also an inherited attribute or a method from a superclass can be *overridden* by the subclass via redefinition.

Multiple inheritance in a class hierarchy occurs when a class is a subclass of two (or more) different classes and hence inherits the attributes and methods of both superclasses from which the subclass inherits may have distinct attributes or methods of the same name, creating an ambiguity. There are several techniques for dealing with ambiguity in multiple inheritance. One solution is to have the system check for ambiguity when the subclass is created, and to let the user explicitly choose which function (attribute or method) is to be inherited at that time. Another solution is to use system default. A third solution is to disallow multiple inheritance if ambiguity occurs.

The concept of providing an interface to objects through which all legal methods are visible (structural encapsulation), and of localizing the implementation of the methods to the objects (behavioral encapsulation) is known as *encapsulation*. Encapsulation enhances implementation independence, modularity and abstraction. To encourage encapsulation, an operation is defined in two parts. The first part, called the *signature* or interface of the operation, specifies the operation name and the arguments. The second part, called the *method* or *body*, specifies the implementation of operation. Operations can be invoked by passing a message to an object, which includes operation name and the parameters. The object then executes the method for that operation. The determination of what implementation is associated with a given method name and class is called *method resolution*. A method is invoked with respect to an object o, and the class to which o belongs determines which implementation to be used. This policy is called *dynamic binding*.

2.2.1 Fuzzy Logic in Object-oriented Database Model

Imprecise information in an object-oriented database can be classified into three categories [78]: imprecision of the data encapsulated in the object, partial class membership, and fuzzy class/superclass membership.

Imprecision of the data: In this category data encapsulated within an object is imprecise. The techniques, similarity measures or possibility distributions, explained in Section 2.1.1 can be used to handle this kind of imprecision.

Partial class membership: This is the imprecision in the membership of an object to a class. This may happen when an object of a class holds a value that is different from the range of values that property can have in the class. An example to this can be given from university departments. Suppose that a computer engineering department gives more emphasis on the faculty members who study on *Theory* or *Database*. In this department a faculty member who studies *AI* will have less membership to the department, say *0.5*.

Fuzzy class/superclass membership: In crisp logic a class is a subclass of another if and only if all objects in the subclass are also a member of the superclass.

In fuzzy logic this relationship changes. If we have two fuzzy classes, the degree of being subclass/superclass changes according to a fuzzy subsethood measure. Consider again the university example. Suppose there is an establishment of a new program called *informatics*, which will be a combination of the programs *mathematics*, *computer engineering* and *industrial engineering*, with the importance levels of *0.1*, *0.9*, *0.5* respectively. The degree of the membership of a faculty member to the new program gets determined by the importance of the faculty member's first level program (either matematics, computer engineering or industrial engineering) within the newly established program together with the membership of the faculty member to its first level program. So, the faculty member that studies *A1* belongs to the new department with a membership of *0.5*0.9=0.45*. Instead of product operator any other conjunction operator, such as *min*, can also be used.

2.3 Aspects of Active Databases

The production rule concept in Expert Systems was extended in the context of active rules, where it is possible to specify database events triggering the rules [66]. The distinguishing feature of an active database system compared to an deductive (passive) one is that it is able to respond automatically to situations (events) that arise inside and outside the database. Active rules can be used to define automated responses to constraint violations, specifying corrective actions or alerting the user of undesirable conditions. Active rules can also be used to monitor events and conditions of specific interest that are not necessarily associated with constraints and to trigger associated actions and notifications within the application. Active rules are especially useful for the dynamic maintenance of integrity constraints as database updates occur. The active behaviour of a database is generally described using rules which most commonly have three components, an event, a condition and an action. A rule with such components is known as an *event-condition-action rule (ECA rule)* [27].

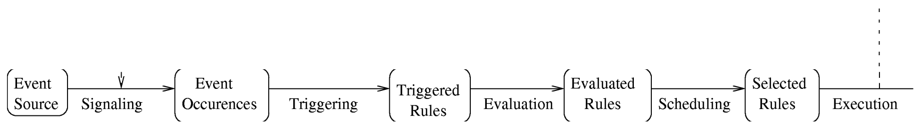


Figure 2.3: Phases of rule execution

- *Events* can be database accesses for retrieval and manipulation (like creation of new facts, updating an attribute, accessing an attribute or invocation of an operation). The user should be able to selectively define events and the system should be responsible for detecting them efficiently.

- Each *condition* is a query formula, or test to be evaluated in the state after the event gets generated but before the activation of the action.

- Each *action* is a chain of one or more procedure calls, which can perform any computation in the database. Actions may update the database, inform the user, or may take some alternative course of actions like causing the creation of a new fact.

An ECA rule lies silent until an occurrence of the event that it is monitoring, when the rule is said to be triggered. The condition of a triggered rule is subsequently evaluated, and if true, then the action of the rule is scheduled for execution.

The type of an event can be primitive or composite. Composite events are raised by some combination of primitive or composite events using a range of event constructors. Most common event constructors are disjunction, conjunction and sequence.

The execution model specifies how a set of rules is treated at run-time. Figure 2.3 illustrates phases of rule execution. Rule execution is affected by the coupling modes. Event-condition coupling mode determines when the condition is evaluated relative to the event that triggers the rule, while condition-action coupling mode indicates when the action is to be executed relative to the evaluation of condition. The options for coupling modes most frequently supported are immediate, deferred and detached. Scheduling phase of rule evaluation determines what happens when multiple rules are triggered at the same time. Rules can be scheduled sequentially or parallel.

2.4 Petri Nets

2.4.1 Regular Petri Nets

A regular Petri net is defined with a structure $(\mathtt{P},\mathtt{T},\mathtt{A},\mathtt{M}_0)$where:

i. `P` is a finite set of *places*.

ii. `T` is a finite set of *transitions*.

iii. $\mathtt{A} \subset (\mathtt{PxT} \cup \mathtt{TxP})$ is a finite set of *arcs* denoting connections between places and transitions.

iv. $\mathtt{M}_0\mathtt{:P}\rightarrow\{0,1,2,...\}$ is the *initial marking* of places.

Petri nets are represented as directed arcs with two type of nodes (places and transitions) connected by arrows that specify the direction of information flow. An arbitrary assignment of tokens to places is called *marking.* A particular marking specifies the state of a system being modeled with a Petri net. In classical Petri nets, a token represents a typeless fragment of information. A black dot symbolizes a single token. Tokens are used to define the execution of a Petri net. Places represent storage for input or output. Transitions represent activities (transformations). Every time an input place of the Petri net is marked, whether ot not the corresponding transition(s) can fire has to be checked. A transition t can fire if and only if all of its input places are marked. The firing of a transition leads to the marking of all output places. Thus, when transition t fires, the subsequent marking satisfies the following equation:

$$M^{'}(p) = \begin{cases} M(p) + 1, & \forall\ p \in \text{output places} \\ M(p) - 1, & \forall\ p \in \text{input places} \\ M(p), & \text{otherwise} \end{cases} \tag{2.4}$$

Figure 2.4 shows what a regular Petri net looks like before and after a transition gets fired.

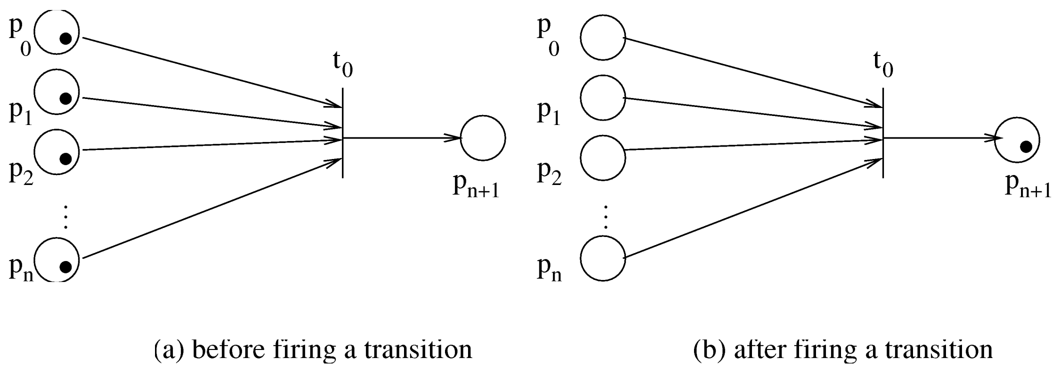


Figure 2.4: Firing a Regular Petri Net

2.4.2 Colored Petri Nets

To facilitate the formal specification and analysis of the structure, information flow, control and computation on systems, Colored Petri Nets (CPNs) have been introduced [54]. Formally a CPN is a structure `(P,T,A,C,CF,E,G)` where:

i. `P,T,A` are the same as those in regular Petri nets.

ii. `C` is a finite set of *data types* (called *colour types*).

iii. `CF: P` $\rightarrow$ `C`, is a *token type (colour) function.* It maps each place to a token type in C.

iv. `E:A` $\rightarrow$ `Expression E(a) of type C(p(a))`, is the *arc function.* It maps each arc to an expression called an arc expression.

v. `G:T` $\rightarrow$ `Boolean expression`, is a *guard function.* It maps each transition to a boolean expression (called a guard expression).

In contrast to regular Petri nets, CPNs can carry complex information. They provide data typing (color sets) and sets of values of a specified type for each place. The expression $E(p,t)$ is the name of a variable associated with the arc from input place p to transition t, and the expression $E(t,p)$ is associated with the transformation performed by transition t on its inputs to produce an output for place p. A guard is a boolean expression on a transition t which must be satisfied before t can fire.

2.4.3 Fuzzy Petri Nets

In FPNs, all components of a Petri net (transitions, places, tokens and arcs) are fuzzified.

Fuzzy Token : The token has a linguistic value, such as *low, medium, high*, defined as membership functions for a linguistic variable. A membership function determines the degree of membership in a particular place, or the truth value of a proposition.

Fuzzy Place : Each place has a predicate or property associated with it. A token in that place is characterized by that property and a level to which it possesses that property or belongs to that place. Fuzzy places hold fuzzy propositions or conclusions.

Fuzzy Transition : This corresponds to *if then* fuzzy production rules and is realized by fuzzy inference algorithms.

Fuzzy Arc : This specifies the required value of a corresponding input/output token. Fuzzy arcs carry fuzzy tokens.

2.4.3.1 Fuzzy Petri Nets for Dealing with Fuzzy Deductive Rules

A FPN is capable of modeling fuzzy deductive rules. These rules are of the type:

IF d_j *THEN* d_k *with Certainty Factor* μ_i

Each place may contain a token associated with a truth value of a proposition, which is quantified as numbers in the unit interval. Each transition is associated with a certainty factor taking values from the unit interval. This model of FPN is defined as a tuple $(P,T,D,I,O,f,\alpha,\beta)$ where

i. P and T are same as before.

ii. $D = \{d_1, ..., d_n\}$ is a finite set of *propositions* .

iii. I: P $\rightarrow$ T is an *input* mapping.

iv. O: T $\rightarrow$ P is an *output* mapping.

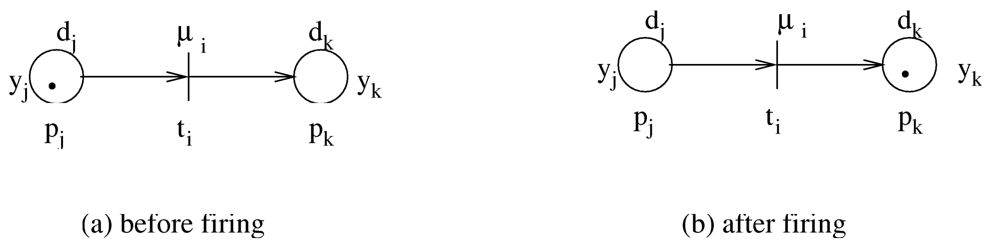


Figure 2.5: Firing a Fuzzy Petri Net

v. `f:` `T` $\rightarrow$ `[0,1]` is a mapping to a *certainty factor* (μ_i) .

vi. α: `P` $\rightarrow$ `[0,1]` is a mapping to a *truth value of a token* .

vii. β: `P` $\rightarrow$ `D` is a mapping from places to propositions.

A token value in place $p_i \in P$ is denoted by $\alpha(p_i) \in [0, 1]$. If $\alpha(p_i) = y_i, y_i \in [0, 1]$ and $\beta(p_i) = d_i$, this then states that the degree of the truth of proposition d_i is y_i. A transition t_i is enabled if $\forall p_i \in I(t_i), \alpha(p_i) \geq \lambda$, where λ is a threshold value in the unit interval. If this transition is fired, then tokens are removed from their input places and a token is deposited onto each of its output places. The truth value of the output tokens are computed through some aggregation function τ, ($y_k = \tau(I(t_i), \mu_i)$, where y_k is $\alpha(O(t_i))$).

Example: The fuzzy deductive rule:

IF d_j *THEN* d_k *with Certainty Factor* μ_i

can be modeled as shown in Figure 2.5. In this example, the truth value of the output token is calculated via the algebraic product, $y_k = y_j * \mu_i$.

2.5 Static Rule Analysis

2.5.1 Termination Analysis

Termination for a rule set is guaranteed if rule processing always reaches a state in which no rule is triggered. Several methods have been proposed in the literature to perform termination analysis. One of them is building a triggering graph by considering the type of triggering events and events generated by the execution of the rule actions [1, 19, 46].

Formally, a *triggering graph (TG)* [1] is a directed graph $\{V, E\}$, where each node in V corresponds to a rule in the rule base and each edge $r_i \rightarrow r_j$ in E means that the action of the rule r_i generates events that trigger r_j. If there are no cycles in the triggering graph, then processing is guaranteed to terminate. TG, however, fails to take into account the details of the interaction between the conditions and the actions of potentially non-terminating rules. That is, although TG has a cycle, it may be the case that, when a rule in the cycle gets triggered, the condition of that rule is not true. As a result, the rule is not executed and the cyclic chain is broken. Consider, for example, the following rule:

```
R1:
  ON update to attribute A of T
  IF new value A > 10
  THEN set A of T to 10
```

The TG for the rule base involving R_1 contains a cycle, as the action of R_1 updates the attribute A of T, which in turn triggers R_1. However, non-termination does not result as the action of R_1 assigns A a value for which the condition of R_1 never becomes true. It is to overcome this limitation in TGs that activation graphs have been introduced.

An *activation graph (AG)* [3] is built upon the semantic information contained in the rule conditions and actions. It is a directed graph $\{V, E\}$, where each node in V corresponds to a rule in the rule base, each edge $r_i \rightarrow r_j$ $(i \neq j)$ in E means that the action of the rule r_i may change the truth value of the condition of r_j from false to true, and each edge $r_i \rightarrow r_i$ means that the condition of r_i may be true after the execution of its own action. If there are no cycles in the AG, then processing is guaranteed to terminate. Some studies [7] rely mostly on the AG while making termination analysis.

Other studies [3] try to detect potential non-termination by using both TG and AG together where a rule set can only exhibit nonterminating behavior when there are cycles in both the triggering and the activation graphs that have at least one rule in common.

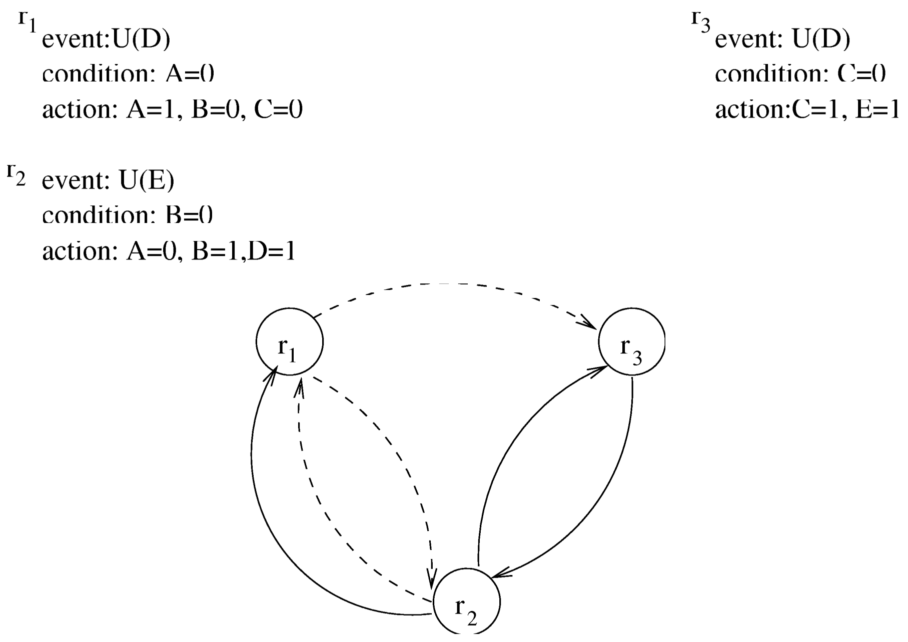


Figure 2.6: A knowledge base and its triggering and activation graph.

Returning to the example given above, the AG for rule R_1 contains no cycle, because its condition can't be true after the execution of its action. Thus, even though the TG contains a cycle, the execution of the rule terminates.

Example: Figure 2.6 illustrates a small knowledge base. The extensional database consists of a single class C having attributes A, B, C, D, E; the type of these attributes is restricted to the set of values $0, 1$. The rule set consists of three rules r_1, r_2, r_3. Rule events are update operations on the attributes, conditions are predicates; actions are sequences of update operations. The TG and AG corresponding to the above rules are illustrated in Figure 2.6. TG arcs are represented by solid lines and AG arcs are represented by dashed lines.

2.5.2 Confluence Analysis

On each execution of the scheduling phase of rule processing, multiple rules may be triggered, and hence be eligible for execution. A rule set is *confluent* if the final state of the database does not depend on which eligible rule has been chosen for execution.

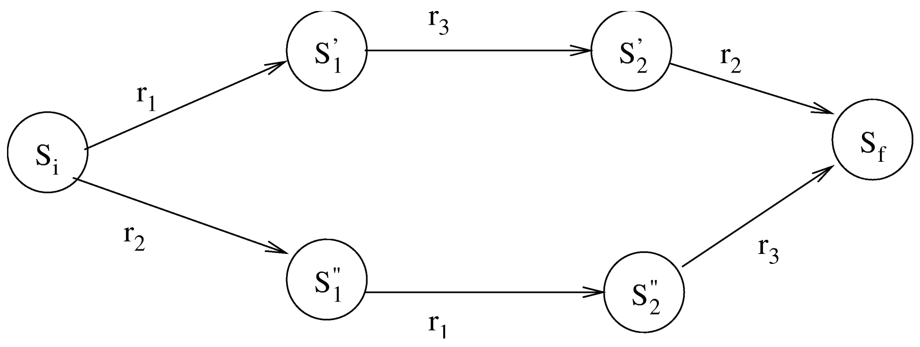


Figure 2.7: Rule execution by means of rule execution sequences

The rule execution process can be described by the notions of *rule execution state* and *rule execution sequence*. Consider a rule set R. A rule execution state S has two components: 1. a database state d, and 2. a set of triggered rules $R_T \subset R$. When R_T is empty, no rule is triggered and the rule execution state is quiescent. A rule execution sequence consists of a series of rule execution states linked by (executed) rules. A rule execution state is complete if its last state is quiescent.

In Figure 2.7, two rule execution sequences for the rule set $R = \{r_1, r_2, r_3\}$ are represented. The initial state S_i is characterized by database state db_i and triggered rule set $R_T = \{r_1, r_2\}$ and the (unique) final state is characterized by a database state db_f and an empty triggered rule set. Each state transition is labeled with the rule whose execution caused the transition.

Confluence can be defined in terms of execution sequences. A rule set is confluent if, for every initial execution state S, every complete rule execution sequence beginning with S reaches the same quiescent state. Then confluence analysis requires the exhaustive verification of all possible execution sequences for all possible initial states. This technique is clearly unfeasible even for a small rule set.

A different approach to confluence analysis is based on the *commutativity* of rule pairs. Two rules r_i and r_j commute if, starting with any rule execution state S, executing r_i followed by r_j produces the same rule execution state as executing r_j followed by r_i; this is shown in Figure 2.8.

If all pairs of rules in a rule set R commute, any execution sequences with

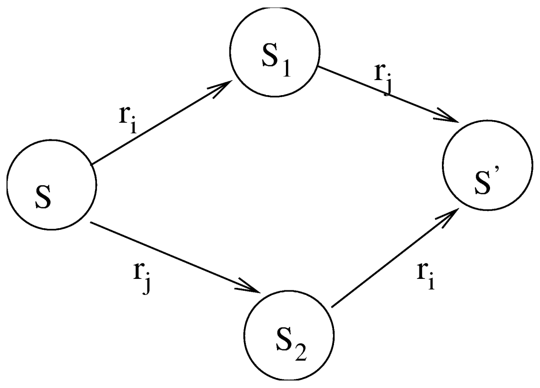


Figure 2.8: Commutativity of a rule pair

the same initial state and executed rules have the same final state. Furthermore, again under the assumption of commutativity, two sequences with the same initial execution state must have the executed rules. Based on these properties, it is possible to state a sufficient condition to guarantee confluence of a rule set: A rule set R is confluent if all pairs of rules in R commute [6].

Confluence may be guaranteed by imposing a total ordering on the active rule set [3]. Consider a prioritized rule set R. If a total ordering is defined on the rules in R, when multiple rules are triggered only one rule at a time is eligible for evaluation. Then, rule processing is always characterized by a single rule execution sequence, which yields a unique final state, and confluence is guaranteed.

2.6 Previous Studies and Contribution of This Study

2.6.1 Active Studies

2.6.1.1 Active Object-Oriented Databases

There have been a couple of projects, such as HIPAC [28], EXACT [29], NAOS [25], Chimera [26], Ode [36], SAMOS [31], Sentinel [21], Reach [82], developing active object-oriented databases. These are all evaluated for a couple of dimensions [62, 63]. Here we only mention three of them, which are Ode [36], Samos [31] and Sentinel [21]. The reason we choose them is that they handle event composition.

In the Ode object-oriented database [36], two different types of rules, constraints and triggers, are associated with the objects. Since these rules are defined at the class level, they can be inherited. Only object updates caused by public member functions are considered as events, and events are not specified explicitly. Constraint and trigger rules defined at [36] are logically different. Constraint rules, which are concerned with the consistency of the object state, affect all instances of a class and are defined permanently. However, triggers are fired whenever the specified conditions become true and are explicitly activated on particular objects. In addition, although constraint actions are executed as part of a transaction violating the object constraints, trigger actions are initiated as separate transactions. For all kinds of rules, only immediate coupling mode is supported in event-condition (EC) coupling.

In SAMOS [31], not only method but also time, transaction and abstract kinds of events are supported. As for the scope of the method events, both the class and object level method events are supported, as the case in Ode. As far as the event consumption mode is concerned, only chronicle one exists. However, all coupling modes are supported (immediate, deferred and detached).

As far as Sentinel [21] is concerned, it supports method, transaction and time events. As the case in both Ode and SAMOS, Sentinel also provides class and object level method events. In Sentinel, some objects are extended with an event interface to enable them to designate some of their methods as primitive event generators. The objects including these event interfaces are termed as reactive objects. If a reactive object detects the occurrence of an event, it signals/informs some other objects called notifiable objects (event and rule objects). These notified objects then take appropriate measures by evaluating conditions and executing actions. Perhaps the most significant feature of Sentinel is that it supports a diverse set of event consumption modes, such as recent, chronicle, continuous and cumulative. For the coupling modes, immediate and deferred modes are supported between event and condition, while only immediate coupling mode is supported between condition and action.

2.6.1.2 Fuzziness in Active Databases

Although the incorporation of fuzziness into active databases introduces much flexibility, only a couple of researchers incorporated fuzziness in their systems. In [14], Bouaziz et al. propose a condition-action (CA) fuzzy trigger , where fuzziness is introduced in the CA part of an ECA rule. In a later work by Bouaziz et al. [16], authors extend the concept of CA trigger to a fuzzy ECA rule by introducing the notion of fuzzy events. In fuzzy ECA rules an event may fire a set of rules. Bouaziz et al. [16] define fuzzy events as fuzzy sets and use linguistic hedges like *high*, *low* and *strong*. Formally, a primitive fuzzy event is represented as a tuple $< e_c, e_f >$ where e_c is a crisp event and e_f is a fuzzy event predicate. When a crisp event is signalled, the value produced upon the operation causing the crisp event is fed into the membership function of e_f. The output of the membership function is called the event match factor, and the fuzzy event is signalled only if the event match factor is greater than zero [16]. Upon the signalling of the fuzzy event, the corresponding rules are fired and their conditions (which are fuzzy predicates on the database) are checked. The action of a rule is started executing depending on the result of condition evaluation.

In a study by Saygin et al. [68], fuzziness is incorporated into rule execution via fuzzy coupling modes and scenarios. Fuzzy primitive events are combined to form fuzzy composite events. In [68], fuzzy concepts are also used in rule scheduling. Saygin et al.'s approach divides the whole set of rules in the system into subsets. Each of those subsets is actually a group of fuzzy sets and represents a particular scenario like emergency or normal. In that study, rules belong to a scenario with a degree of membership of that scenario. Saygin et al. [68] also study similarity-based event detection.

2.6.1.3 Our Study vs Active Studies

Our study differs from the previous research efforts mentioned above in a number of respects. First, the active object-oriented database systems Ode [36], Samos [31] and Sentinel [21] aim only at designing a full-fledged active database system and they do not consider the intelligent behaviour required to handle deductive queries. However, in our study we not only integrate active rules but also deductive rules so that the intelligent behaviour required to handle deductive queries is also an integrated part of objects. Second, we consider the uncertainty and vagueness inherent in knowledge-intensive applications in defining data, attributes, objects, classes, rules (either ECA rules or deductive rules) and in their inference mechanisms. On the other hand, active object-oriented database systems [36, 31, 21] assume crisp domains and ignore uncertainty in their applications. That is, attributes, objects, classes and rules are defined with a high degree of crispness. Third, although Bouaziz et al. [14, 15, 16] introduce fuzziness in their ECA rules via using linguistic terms and provide a fuzzy inference mechanism, the way they associate fuzzy events with fuzzy CA rules in the inference mechanism is different from ours. That is, they use a technique called squeezing to associate fuzzy events with fuzzy CA rules by squeezing the result of the fuzzy CA rule set with the event match factor. If the event match factor is lower, there is a more movement towards less significant actions. However, in our approach the effect of fuzzy event on fuzzy inference is smoother, i.e. the action chosen at the end still comes from the possible actions before the effect of event is applied. The details of our fuzzy inference mechanism is given in Section 3.2. Fourth, we aim at modeling the detection and inference of fuzzy composite events, which to the best of our knowledge has not yet been studied before. Although Bouaziz et al. [16] study fuzzy events, they do not give a general model of fuzzy events that capture event composition. On the other hand, although the active object-oriented databases mentioned above model composite event detection, (Ode [36] uses finite state automata, SAMOS [31] uses colored petri-nets, and Sentinel [21] uses query graphs for this purpose), none of them considers fuzziness. Finally, in our study, we develop a model for the membership of fuzzy rules to the scenarios, which we use in fuzzy inference during the calculation of event match factor. However, in the study by Saygin et al. [68], although scenario concept is introduced, the way of calculating membership values of fuzzy rules to the scenarios is not specified.

2.6.2 Fuzzy Petri Nets and Static Rule Analysis

2.6.2.1 Fuzzy Petri Nets

There have been a number of studies using FPNs. Chen et al. [23] consider a representation of fuzzy production rules with certainty factors. The reasoning algorithm used in that study determines whether there exists an antecedent-consequence relationship between two propositions. If this is the case, the degree of truth of the consequent proposition is evaluated from that of antecedent propositions. In a later work by Chen et al. [22], the authors extend their previous work with a Weighted Fuzzy Petri Net (WFPN) model, where certainty factors, truth values of propositions and weights of propositions are represented by fuzzy numbers. However, in all their work [23, 22] only exact matching is allowed. A similar approach taken by Manoj et al. [55] modify Chen et al.'s [23] fuzzy reasoning algorithm after finding out that it does not work with all types of data. Chun et al. [24] study algebraic forms of a state equation of the FPN, which are systematically derived using a matrix representation. They use state equations to perform both forward and backward reasoning. However, they change the firing rule of conventional Petri nets. A different FPN model is studied by Bugarin et al. [18], who present the execution of a knowledge base by using a data driven strategy based on the sup-min compositional rule of inference. However, since the arrangement of the linking transitions and the applied algorithm depend on the initial marking, their work is not appropriate for large systems. Finally, Scarpelli et al. [67] propose a High Level Fuzzy Petri Net (HLFPN) for modeling fuzzy reasoning based on compositional rule of inference. Their forward reasoning algorithm consists of the extraction of a subnet from an entire net. However, after extracting the subnet, it is not allowed to have concurrent inference.

2.6.2.2 Static Rule Analysis

While TG arcs are syntactically derivable, it is very difficult to determine precisely the arcs of an AG. In Baralis et al. [3], it is assumed that conditions and actions are both represented by relational algebra expressions. Unfortunately, for the host languages that are not based on relational algebra or relational calculus, it is difficult to infer the truth value of a condition from the imperative code of a rule's action [30]. Due to this fact, most of the studies [30, 46, 74, 58] including ours consider only triggering graph information during termination analysis.

Most of the existing studies on rule analysis only deal with simple rule languages; i.e., languages that support only a limited set of constructs for specifying active behavior. If the rules become complex (like having composite events, supporting complex conditions/actions, etc), their analysis also becomes complex. The reason is that there are additional elements on which the triggering of a rule depends. For example, a rule defined on a complex event is triggered only when all component events occur. Therefore, compared to a simple rule language, it is more difficult to decide when rules may generate an infinite loop during execution. That is, in a system having only primitive events, an edge in the triggering graph indicates that one rule can generate an event that can in turn trigger another. However, when a rule r has a composite event, it may be that no other single rule in the rule base can trigger r, but that a subset of the rules together may be able to trigger r. Vaduva et al. [74] and Dinn et al. [30] consider compositions while performing termination analysis.

2.6.2.3 Our Study vs Other Studies on FPNs

In this study, we propose an extended Fuzzy Petri Net for modeling fuzzy rule-based reasoning. Compared to the previous studies existing in literature, we add a couple of new features to FPNs. First, we can model any rule type (active or deductive), unlike the previous works, which do not model active rules. Second, we not only model the composition of conditions but also the composition of events. In contrast, since previous studies using FPNs cannot model active rules, they do not consider event composition. Additionally, we extend the functionalities of transitions. That is, we render them capable of performing fuzzification, event/condition composition, concurrent execution, and combination in addition to the sup-min composition. Fourth, since we use the features of Colored Petri Nets, which support a rich set of token types, we provide parameter passing through the FPN. This provides values of conditions and actions to be calculated from the parameters of events. Fifth, we check the properties of our system using the FPN structure. This is not considered by any other study that utilizes FPNs. Our FPN structure already contains the Triggering Graph information and supports the static analysis of the rule base. That is, we can perform the termination analysis easily by just using the FPN. Therefore, there is no need to do extra work to construct Triggering Graph as required by other studies [1, 19, 30, 46, 74], which use structures other than FPN for studying rule analysis. In addition, while performing termination analysis, we also handle event and condition compositions, and eliminate false cycles. Finally, we guarantee confluent rule execution with our fuzzy inference algorithm that we introduce in Chapter 5.

CHAPTER 3

INCORPORATING FUZZINESS INTO ACTIVE RULES

In this chapter, we describe how we incorporate fuzziness into active rules. In Section 3.1, we explain the scenario concept. In Section 3.2, we show how we perform fuzzy inferences in an active rule-base. This is followed by our model for the similarity of a rule to the scenario in Section 3.3. Finally in Section 3.4, we describe our approach for handling fuzziness in active rules in an application, which is the alarm treatment in an industrial drive control system.

3.1 Inter-rule Fuzziness via Scenarios

A *scenario* is a set of rules of a database state. We partition the rule space and call each partition a *scenario*. There can be only one active scenario at a time. Switching between the scenarios is determined by the user. During the fuzzy inference cycle, each rule has a firing threshold which is used to decide if a rule will be fired or not. For that, we calculate the strength of events for the rules fired. This calculation is affected by the current active scenario. If the result is greater than or equal to the threshold value, then the rule is fired.

The concept of scenarios on rules is studied in [68]. In that study, strength of an event e, for rule r within the scenario s is calculated using the formula:

$$strength < e, r, s >= \mu_s(r) * \mu_{e_f}(value(e_c)) \tag{3.1}$$

which uses scalar multiplication, where $value(e_c)$ is the value of the event (either fuzzy or crisp) detected, μ_{ef} is the membership function of the fuzzy event e_f and $\mu_s(r)$ is the similarity of the rule r to the current scenario s.

In [68] how the membership functions of scenarios for the fuzzy rules in the system found is unclear; it is only said that this can be determined according to the results of priori simulations. However, in Section 3.3 we develop a model for membership calculations of fuzzy rules to the scenarios.

3.2 Incorporating Fuzzy Inference into Active Rules

Since a fuzzy rule-based system consists of a set of fuzzy rules with partially overlapping antecedents, a particular input to the system often "triggers" multiple fuzzy rules (more than 1 (one) rule matches the input to a nonzero degree). In those cases, we need a fuzzy inference mechanism. At this point, it is appropriate to define the nature of antecedents and consequents of a fuzzy active rule. Antecedent part is constructed from the events and conditions, whereas the consequent part is constructed from the actions. Fuzzy inference becomes very important in the cases where the activation of an event fires a couple of rules whose action part do the same thing but with different fuzzy linguistic terms. These rules are grouped to evaluate in case of a suitable event invocation. If the event is invoked, there needs to be a mechanism for determining the total outcome (or action) of these rules. It is more appropriate to take a unique action (which is the most suitable action for the current situation) from the possible actions list instead of executing each action one by one.

We define our rules, which can be either active or deductive type, in the following form:

```
< ON event list <event threshold> >
IF condition list <EC coupling>
THEN action/conclusion <CA coupling>
```

where the parts inside the <> means they are optional. If the ON part is omitted, it becomes a deductive rule. We use the abstract kind of events for the deductive rules so that internally all rules are of active type. If a rule defined is an active one and event threshold is omitted, then exact matching with a value of 1(one) is assumed. If coupling modes are omitted in an active rule, they are assumed as immediate. Here all coupling modes are assumed as immediate.

The structure of concurrent active rules are as follows:

$\mathtt{R}_1$: ON $\mathtt{x}_1$ is $\mathtt{A}_{11}$ AND/OR $\mathtt{x}_2$ is $\mathtt{A}_{12}$ $< event\_threshold_{R_1} >$
IF $\mathtt{y}_1$ is $\mathtt{B}_{11}$ AND/OR $\mathtt{y}_2$ is $\mathtt{B}_{12}$
THEN z is $\mathtt{C}_1$

$\mathtt{R}_2$: ON $\mathtt{x}_1$ is $\mathtt{A}_{21}$ AND/OR $\mathtt{x}_2$ is $\mathtt{A}_{22}$ $< event\_threshold_{R_2} >$
IF $\mathtt{y}_1$ is $\mathtt{B}_{21}$ AND/OR $\mathtt{y}_2$ is $\mathtt{B}_{22}$
THEN z is $\mathtt{C}_2$

....

$\mathtt{R}_n$: ON $\mathtt{x}_1$ is $\mathtt{A}_{n1}$ AND/OR $\mathtt{x}_2$ is $\mathtt{A}_{n2}$ $< event\_threshold_{R_n} >$
IF $\mathtt{y}_1$ is $\mathtt{B}_{n1}$ AND/OR $\mathtt{y}_2$ is $\mathtt{B}_{n2}$
THEN z is $\mathtt{C}_n$

If the facts for which we are doing inferencing are x_{10}, x_{20} for event, and y_{10}, y_{20} for condition, we perform the following fuzzy inference steps to find the appropriate action:

Fuzzy Event Matching:

In this step, matching factor for the event part of the rules gets calculated using the *strength of the event* formula given by Equation 3.1. Before that, we fuzzify the event, i.e., calculate the value of $\mu_{e_f}(value(e_c)))$ in order to determine the membership value of the fuzzy event detected. For that either min (for conjunctions of events) or max (for disjunctions of events) operator is used.

$$strength < e, r, s >= \mu_s(r) * \mu_{e_f}(value(e_c))$$

where

$$\mu_{ef}(value(e_c)) = \mu_{A_{i1} and/or A_{i2}}(x_{10}, x_{20}) = min/max(\mu_{A_{i1}}(x_{10}), \mu_{A_{i2}}(x_{20})) \quad (3.2)$$

If *strength<e,r_i,s>* $\geq$ *event_threshold$_{r_i}$*, the rule r_i gets fired.

Fuzzy Condition Matching:

In this step, matching factor for the condition part of the rules gets calculated using *min* operator for the conjuntion of conditions and *max* operator for the disjunction of conditions.

$$\mu_{B_{i1} and/or B_{i2}}(y_{10}, y_{20}) = min/max(\mu_{B_{i1}}(y_{10}), \mu_{B_{i2}}(y_{20})) \quad (3.3)$$

If $\mu_{B_{i1} and/or B_{i2}}(y_{10}, y_{20}) > 0$, we go to the next step.

Fuzzy Inference:

Total value of the matching factor for antecedent (event and condition together) of each rule gets calculated using the product operator.

$$\mu_{A_{r_i}} = product(strength < e, r_i, s >, \mu_{B_{i1} and/or B_{i2}}(y_{10}, y_{20})) \quad (3.4)$$

Then, we do clipping on the action part of each rule according to this fuzzy antecedent matching value.

$$\mu_{C_i'}(z) = min(\mu_{A_{r_i}}, \mu_{C_i}(z)) \quad (3.5)$$

Combination:

This is done by superimposing all fuzzy conclusions about an action. This is based on applying the max fuzzy disjunction operator to multiple possibility distributions of the output variable.

$$\mu_{C'}(z) = max(\mu_{C_1'}(z), ..., \mu_{C_n'}(z)) \quad (3.6)$$

3.3 A model for the Similarity of the Rule to the Scenario

Any active rule can be written as:

$$C : -A_1, A_2, ..., A_n \tag{3.7}$$

which is equivalent to

$$A_1 \wedge A_2 \wedge ... \wedge A_n \longrightarrow C \tag{3.8}$$

where antecedent (A_i's) is composed of events and conditions, and consequent (C) is an action. In order to determine whether a rule can be fired within the current scenario, similarity of the rule to the scenario rules should be calculated. We use the following formula while evaluating the similarity of any rule R_i in the rule set to the current scenario S:

$$\mu_s(r) = max(|min(max(\mu(A_s, A_r)), max(\mu(C_s, C_r))) * RLV_{rs}/RLV_{max}|) \tag{3.9}$$

where $A_s \in S$, $\forall A_r \in R_i$, $C_s \in S$, $\forall C_r \in R_i$, RLV_{rs}, $RLV_{max} \in S$. A_s is the antecedent of a current scenario meta rule and A_r is the antecedent of R_i (the rule to be evaluated), C_s is the consequent of a meta rule in the scenario and C_r is the consequent of the rule to be evaluated, RLV_{rs} is the relevance value of the meta rule to the current scenario and RLV_{max} is the maximum of those relevance values.

3.4 Example - An Overheating Alarm Generation in a Drive System of a Couple of Motors

We take this example rules from Bouaziz et al. [16]. It is an application of how to generate an alarm notification in a system of more than 1(one) motor where there may be a motor overheating. Here is the set of rules grouped for this inference:

```
R1:
    ON there is a change on any motor temperature
       AND motor.temperature is hot 0.1
    IF some motors hot AND  most deltas big_positive
    THEN   alarmnotification is low
R2:
    ON there is a change on any motor temperature
       AND motor.temperature is hot 0.1
    IF some motors very_hot AND  some deltas big_positive
    THEN alarmnotification is low
R3:
    ON there is a change on any motor temperature
       AND motor.temperature is hot 0.1
    IF some motors very_hot AND  most deltas big_positive
    THEN alarmnotification is medium
R4:
    ON there is a change on any motor temperature
       AND motor.temperature is hot 0.1
    IF most motors hot AND  some deltas big_positive
    THEN alarmnotification is medium
R5:
    ON there is a change on any motor temperature
       AND motor.temperature is hot 0.1
    IF most motors hot AND  most deltas big_positive
    THEN alarmnotification is high
R6:
    ON there is a change on any motor temperature
       AND motor.temperature is hot 0.1
    IF most motors very_hot
    THEN   alarmnotification is high
```

On the above rule set, the event generated is the update on the motor temperature and the new value of the temperature being hot and conditions check the delta differences (temperature difference divided by the previous temperature) and action parts produce alarm notification with different linguistic terms.

Assume that in the drive system of motors we have emergency scenario with the following meta_rules (protoforms):

```
Remergency1:
            ON there is a change on any motor temperature
               AND motor.temperature is hot
            IF most motors hot AND  most deltas big_positive
            THEN alarmnotification is high
Remergency2:
            ON there is a change on any motor temperature
               AND motor.temperature is hot
            IF most motors very_hot
            THEN alarmnotification is high
```

According to the above meta rules, emergency scenario is $R_{emergency1} \vee R_{emergency2}$. This means that any rule similar to either $R_{emergency1}$ or $R_{emergency2}$ can be fired when the emergency scenario is on. So what needs to be calculated is the similarities of the rules in the system to the meta_rules in the emergency scenario.

Assume that we have two motors. At time t_1 for the motors m_1 and m_2 the temperatures measured were 62.5^oC and 160^oC and at time t_2 they are measured as 125^oC and 160^oC respectively. Assume also that for the active rule evaluation following membership functions and similarity relations are used:

temperature:
normal trapezoidal(0,0,120,140)
hot trapezoidal(120,140,300,300)
very_hot trapezoidal(145,160,300,300)

delta:
negative trapezoidal(-1,-1,-0.4,-0.2)
big_negative trapezoidal(-1,-1,-0.8,-0.6)
small_negative trapezoidal(-0.8,-0.6,-0,4,-0.2)

zero trapezoidal(-0.4,-0.2,0.2,0.4)

small_positive trapezoidal(0.2,0.4,0.6,0.8)

big_positive trapezoidal(0.6,0.8,1,1)

positive trapezoidal(0.2,0.4,1,1)

alarmseverity:

zero trapezoidal(0,0,0.5,1.0)

low trapezoidal(0.5,1.0,1.5,2.0)

medium trapezoidal(1.5,2.0,2.5,3.0)

high trapezoidal(2.5,3.0,4.0,4.0)

fuzzy quantifiers:

few triangular(0,0.2,0.3)

some triangular(0.2,0.45,0.7)

most triangular(0.6,0.8,1)

Table 3.1: Similarity Matrix for Temperature

Temperature	normal	hot	very_hot
normal	1	0.4	0
hot	0.4	1	0.9
very_hot	0	0.9	1

Table 3.2: Similarity Matrix for AlarmNotification

Alarmnotification	zero	low	medium	high
zero	1	0.7	0.5	0
low	0.7	1	0.7	0.5
medium	0.5	0.7	1	0.7
high	0	0.5	0.7	1

Fuzzy Event Matching Phase:

Fuzzify the event:

After we fuzzify the event, we find $\mu_{hot}(125) = 0.25$. Since $\mu_{hot}(125) > 0$ all the rules get fired.

Table 3.3: Similarity Matrix for Quantifiers

quantifier	few	some	most
few	1	0.5	0
some	0.5	1	0.5
most	0	0.5	1

Calculate the strength of event and check it with the rule threshold:

For this $\mu_s(r_i)$ values for the rules are calculated using the Formula 3.9. Also assume that following relevance values are used:

RLV(R$_{emergency1}$,Emergency) = 1, RLV(R$_{emergency2}$,Emergency) = 1.5

Using the similarity matrices, emergency scenario meta_rules and relevance values, $\mu_{emergency}(r_1)$, i.e. relatedness of R$_1$ to the emergency scenario, is calculated as follows:

```
Max[Min(μsim(hot,hot), μsim(most,some),
        μsim(hot,hot), μsim(most,most),
        μsim(big_positive,big_positive),
        μsim(high,low)) * 1/1.5,
    Min(μsim(hot,hot), μsim(most,some),μsim(very_hot,hot),
        μsim(high,low)) *1.5/1.5]=
Max[Min(1, 0.5, 1, 1, 1, 0.5) * 1/1.5,
    Min(1, 0.5, 0.9, 0.5) * 1.5/1.5] =
Max(0.5*1/1.5 , 0.5*1.5/1.5) =
Max(0.33, 0.5) = 0.5
```

Applying the same approach following values are calculated for the similarities of the rules to the current scenario *emergency*:

$\mu_{emergency}(R_2)$=0.5, $\mu_{emergency}(R_3)$=0.5

$\mu_{emergency}(R_4)$=0.7, $\mu_{emergency}(R_5)$=0.9

$\mu_{emergency}(R_6)$=1

and using the Equation 3.1, event strengths are calculated as:

for R_1 ; 0.5 * $\mu_{hot}(125)$ = 0.5 * 0.25 = 0.125

for R_2 ; 0.5 * $\mu_{hot}(125)$ = 0.5 * 0.25 = 0.125

for R_3 ; 0.5 * $\mu_{hot}(125)$ = 0.5 * 0.25 = 0.125

for R_4 ; 0.7 * $\mu_{hot}(125)$ = 0.7 * 0.25 = 0.175

for R_5 ; 0.5 * $\mu_{hot}(125)$ = 0.9 * 0.25 = 0.225

for R_6 ; 1.0 * $\mu_{hot}(125)$ = 1.0 * 0.25 = 0.25

Threshold values for the rules are given as 0.1, therefore all the rules succeed.

Fuzzy Condition Matching Phase:

Evaluate R_1:

"Motors are hot" with degrees 0.25 and 1. Their average is 0.625. "Some motors are hot" evaluates to $\mu_{some}(0.625) = 0.3$. Delta values are 1 ((125-62.5)/62.5) and 0 respectively. "Deltas are big positive" with degrees $\mu_{big\_positive}(1) = 1$ and $\mu_{big\_positive}(0) = 0$. Their average is (1 + 0)/2= 0.5. "Most deltas are big positive" evaluates to $\mu_{most}(0.5) = 0$. "Some motors are hot and most deltas are big positive" evaluates to min(0.3,0) = 0. Since condition match factor is 0, R_1 `fails`.

Evaluate R_2:

"Motors are very_hot" with degrees 0 and 1. Their average is 0.5. "Some motors are very_hot" evaluates to $\mu_{some}(0.5) = 0.8$. Delta values are 1 ((125-62.5)/62.5) and 0 respectively. "Deltas are big positive" with degrees $\mu_{big\_positive}(1) = 1$ and $\mu_{big\_positive}(0) = 0$. Their average is (1 + 0)/2= 0.5. "Some deltas are big positive" evaluates to $\mu_{some}(0.5) = 0.8$. "Some motors are very_hot and some deltas are big positive" evaluates to min(0.8,0.8) = 0.8. Since condition match factor is 0.8, R_2 `succeeds`.

Evaluate R_3:

"Motors are very_hot" with degrees 0 and 1. Their average is 0.5. "Some motors are very_hot" evaluates to $\mu_{some}(0.5) = 0.8$. Delta values are 1 ((125-62.5)/62.5) and 0 respectively. "Deltas are big positive" with degrees $\mu_{big\_positive}(1) = 1$ and $\mu_{big\_positive}(0) = 0$. Their average is (1 + 0)/2= 0.5. "Most deltas are big positive" evaluates to $\mu_{most}(0.5) = 0$. "Some motors are very_hot and most deltas are big positive" evaluates to min(0.8,0) = 0. Since condition match factor is 0, R_3 `fails`.

Evaluate R_4:

"Motors are hot" with degrees 0.25 and 1. Their average is 0.625. "Most motors are hot" evaluates to $\mu_{most}(0.625) = 0.125$. "Some deltas are big positive evaluates to $\mu_{some}(0.5) = 0.8$. "Most motors are hot and some deltas are big positive" evaluates to min(0.125,0.8) = 0.125. Since condition match factor $0.125 > 0$, `R`$_4$ `succeeds`.

Evaluate R_5:

"Motors are hot" with degrees 0.25 and 1. Their average is 0.625. "Most motors are hot" evaluates to $\mu_{most}(0.625) = 0.125$. "Most deltas are big positive evaluates to $\mu_{most}(0.5) = 0$. "Most motors are hot and most deltas are big positive" evaluates to min(0.125,0) = 0. Since condition match factor is 0, `R`$_5$ `fails`.

Evaluate R_6:

"Motors are very_hot" with degrees 0 and 1. Their average is 0.5. "Most motors are very_hot" evaluates to $\mu_{most}(0.5) = 0$. "Most motors are very_hot" evaluates to min(0) = 0. Since condition match factor is 0, `R`$_6$ `fails`.

Fuzzy Inference:

Upto this point only R$_2$ and R$_4$ succeed. Their antecedent matching degrees are
for R$_2$, product(0.125,0.8) = 0.1.
for R$_4$, product(0.175,0.125) = 0.022.
So clipping value of 0.1 is used for the action part of R$_2$, which is an alarm notification of `low` and clipping value of 0.022 is used for the action part of R$_4$, which is an alarm notification of `medium` .

Combination:

Maximum of the clipping values are taken, since we have R$_2$ and R$_4$, max(0.1,0.022) = 0.1. This means an alarm notification of `low` needs to be sent.

CHAPTER 4

FUZZY PETRI NETS FOR MODELING FUZZY RULE-BASED REASONING

We use Fuzzy Petri Nets to represent the knowledge and model the behavior of the system. In this Chapter, we present our FPN model for fuzzy rule-based reasoning. More specifically, we show how we map the fuzzy rules to FPN and how we construct the FPN. We give two example applications; alarm treatment in an industrial drive control system [16], and selection of the washing cycle in a washing machine [78].

4.1 Fuzzy Petri Nets for Fuzzy Rules

We introduce the following Fuzzy Petri Net (FPN) structure to model the fuzzy rules; (P,P_s,P_e,T, TF, TRTF, A, I, O, TT, TTF, AEF, PR, PPM, TV) where:

i. P is a finite set of *fuzzy places*. Each place has a property associated with it, in which

   - $P_s \subset P$ is a finite set of *input places* for primitive events.

   - $P_e \subset P$ is a finite set of *output places* for actions or conclusions.

   Each place is identified with a unique *place id*.

ii. `T` is a finite set of *fuzzy transitions.* Each transition is identified with a unique *transition id.*

iii. `TF` is a finite set of *transition functions*, which perform activities of fuzzy inference.

iv. `TRTF:` `T` $\rightarrow$ `TF` is *transition type function*, mapping each transition $\in T$ to a transition function $\in TF$.

v. `A` $\subset$ `(PxT` $\cup$ `TxP)` is a finite set of *arcs* for connections between places and transitions, where

- `I:` `P` $\rightarrow$ `T` is an *input* mapping.
- `O:` `T` $\rightarrow$ `P` is an *output* mapping.

vi. `TT` is a finite set of *fuzzy token (color) types.* Each token has a linguistic value (i.e. low, medium and high), which is defined with a membership function.

vii. `TTF:P` $\rightarrow$ `TT` is *token type function*, mapping each fuzzy place $\in P$ to a fuzzy token type $\in TT$. A token in a place is characterized by the property of the place and a level to which it possesses that property.

viii. `AEF:` `Arc` $\rightarrow$ `expression`, is *arc expression function* mapping each arc to an expression, which carries the information (token value).

ix. `PR` is a finite set of *propositions*, corresponding to either events or conditions or actions/conclusions.

x. `PPM:` `P` $\rightarrow$ `PR`, is a *fuzzy place to proposition mapping*, where | PR |=| P |.

xi. `TV:` `P`$\rightarrow$ `[0,1]` is *truth values of tokens* (μ_i) assigned to places. It holds the degree of membership of a token to a particular place.

A token value in place $p_i \in P$ is denoted by $TV(p_i) \in |0,1|$. If $TV(p_i)$=μ_i, $\mu_i \in |0,1|$ and $PPM(p_i)$=d_i. This states that the degree of the truth of proposition d_i is μ_i. A transition t_i is enabled if $\forall\ p_i \in I(t_i)$, $\mu_i > 0$. If this transition t_i is fired,

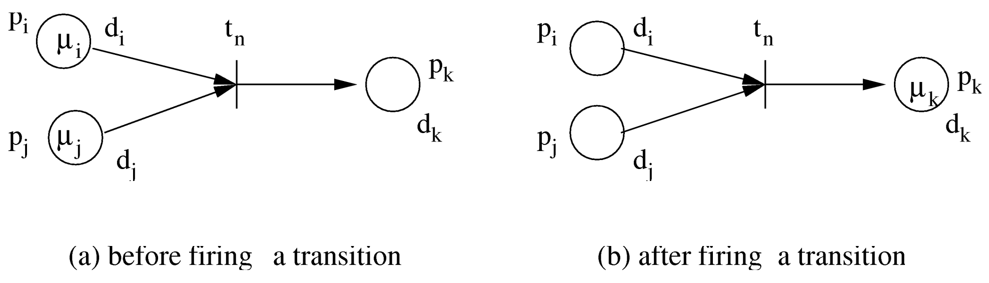


Figure 4.1: Firing the Fuzzy Petri Net.

tokens are removed from input places $I(t_i)$ and a token is deposited onto each of the output places $O(t_i)$. Since we provide parameter passing, the token value of an output place $p_k \in O(t_i)$ is calculated from that of the input places $I(t_i)$ using the transition function TF_i, where TF_i=$TRTF(t_i)$. This token's membership value to the place p_k, (i.e μ_k=$TV(p_k)$), is part of the token and gets calculated within the transition function TF_i, where μ_k= $TF_i(I(t_i))$.

Example: The fuzzy deductive rule *IF d_i and d_j THEN d_k* can be modeled as shown in Figure 4.1. In this example, $PPM(p_i)$=d_i, $PPM(p_j)$=d_j, $PPM(p_k)$=d_k, $TV(p_i)$=μ_i and $TV(p_j)$=μ_j. Suppose that $\mu_i > 0$ and $\mu_j > 0$. This means that transition t_n is enabled and fired. Tokens are removed from $I(t_i)$, which are p_i and p_j, and deposited onto $O(t_i)$, which is p_k. Suppose that the transition function of t_n, which is $TF_n = TRTF(t_n)$, is defined as an algebraic product function. Then the truth value of the output token (membership degree) is calculated as $TV(p_k)$=$TF_n(I(t_n))$=μ_i*μ_j=μ_k.

4.1.1 Mapping Fuzzy Rules to Fuzzy Petri Net

We define our rules, which can be either active or deductive type, in the following form:

```
Ri:
   < ON ei <event threshold> >
   IF ci
   THEN ai
```

where e_i corresponds to either a primitive or a composite event. If it is a composite one, it can be constructed from conjunctions and/or disjunctions of primitive/composite events. The same applies to c_i, which can either be a primitive or a composite condition. On the other hand, the action/conclusion part a_i can only contain a primitive action.

We place the constituents of our rules onto the FPN places and transitions. Here is the list of items that we consider while doing this mapping:

1. `PR`=$\{$`pr`$_1$,`pr`$_2$, ..., `pr`$_n\}$ is a finite set of propositions. A proposition can be any of the following:

   a. primitive events,

   b. fuzzy primitive events,

   c. fuzzy composite events (conjunctions/disjunctions of fuzzy primitive/composite events),

   d. similarities of rules to the current scenario,

   e. fuzzy primitive conditions,

   f. fuzzy composite conditions (conjunctions/disjunctions of fuzzy primitive/composite conditions),

   g. clipped actions/conclusions, and

   h. for each concurrent rule set:

      - a combined action, and
      - a fuzzy action.

2. `PPM: P` $\rightarrow$ `PR`, is a mapping from places to propositions where $|PR| = |P|$. For each proposition pr_i, i=1, ..., n, let p_i be a place in the FPN. Some of these places are specialized as P_s and P_e.

   a. `P`$_s \subset$ `P` is a finite set of input places corresponding to primitive events. They are the starting places in the fuzzy inference mechanism.

b. $\texttt{P}_e \subset \texttt{P}$ is a finite set of output places, which correspond to fuzzified actions/conclusions. They are the final places in the fuzzy inference mechanism.

3. $\texttt{T=}\{\texttt{t}_1\texttt{,t}_2\texttt{, ..., t}_m\}$ is a finite set of transitions. While executing transitions, we go through the inference steps. Transitions use the values provided by input places and produce values for output places by using the specific purpose functions. Their functionalities can be listed as:

   a. fuzzification of primitive events,

   b. fuzzification of primitive conditions,

   c. providing fuzzy event/condition composition (conjunctions/disjunctions of fuzzy primitive/composite events/conditions),

   d. providing a pass from event to condition evaluation:

      - calculates strengths of events,
      - checks the rule threshold,
      - provides input for the condition evaluation,

   e. finding clipping values for the individual rules by using the event and condition match factors of each rule,

   f. combining the outcome of concurrent actions/conclusions, and

   g. determining the overall fuzzy action,

   h. triggering events (marking primitive event places).

4. $\texttt{A} \subset \texttt{(PxT} \cup \texttt{TxP)}$ is a finite set of arcs for connections between places and transitions. Connections between the input places and transitions (PxT) and connections between the transitions and output places (TxP) are provided by arcs. Arcs can be expressed in matrix forms. For this, output and input incidence matrices are used. Since the number of arcs between a transition and a place is either 1 or 0, both matrices are binary matrices.

5. $\texttt{AEF:A} \rightarrow \texttt{expression}$, is a mapping for each arc to an expression, which can be one of the following:

   a. an input arc expression for arcs $\in PxT$, consisting of the variables of the input place of the arc, which carries information of an input place token.

   b. output arc expression for arcs $\in TxP$ consisting of variables to be assigned to output places.

6. `TV:P` $\rightarrow$ `[0,1]` is a mapping for the truth values of propositions (μ) assigned to places. It can be any of the following (only one of them for each place):

   a. 1 (one) for the crisp event,

   b. truth values of the fuzzy primitive events/conditions,

   c. truth values of the composite events/conditions,

   d. similarities of rules to the current scenario,

   e. strengths of events,

   f. clipping values for actions/conditions,

   g. 1 (one) for the combined actions, and

   h. a truth value for the overall fuzzy action/conclusion.

7. `TT` is a finite set of token(color) types.

8. `TTF:P` $\rightarrow$ `TT` is a token type function, mapping each place $\in P$ to a token type $\in TT$.

Figure 4.2 shows how we realize the steps of fuzzy inference using the FPN structure.

4.1.2 Constructing the FPN

First the rule definitions are obtained from the user. For each rule, a rule object is created, and the event, condition and action parts of a rule are examined. While doing this, FPN places are created. Then, the fuzzy inference groups, which are the concurrent rule sets that get triggered at the same time, are determined.

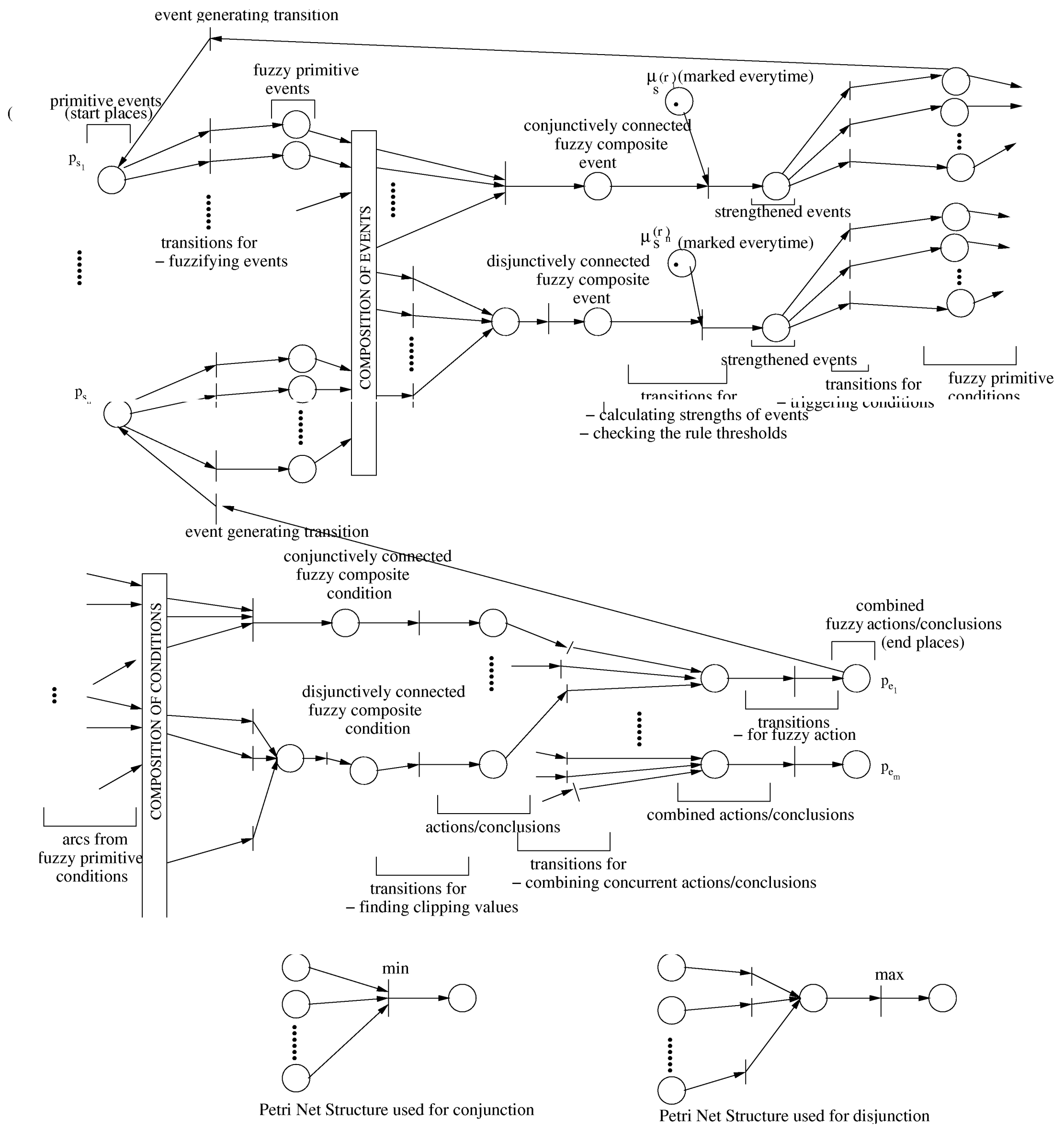

event generating transition
primitive events
(start places)
fuzzy primitive events
p_{s_1}
transitions for
– fuzzifying events
COMPOSITION OF EVENTS
$\mu_s(r)$ (marked everytime)
conjunctively connected fuzzy composite event
strengthened events
$\mu_{s_n}^{(r)}$ (marked everytime)
disjunctively connected fuzzy composite event
strengthened events
transitions for
– calculating strengths of events
– checking the rule thresholds
transitions for
– triggering conditions
fuzzy primitive conditions
p_{s_n}
event generating transition
conjunctively connected fuzzy composite condition
COMPOSITION OF CONDITIONS
disjunctively connected fuzzy composite condition
combined fuzzy actions/conclusions
(end places)
p_{e_1}
transitions
– for fuzzy action
p_{e_m}
combined actions/conclusions
actions/conclusions
arcs from fuzzy primitive conditions
transitions for
– combining concurrent actions/conclusions
transitions for
– finding clipping values
min
Petri Net Structure used for conjunction
max
Petri Net Structure used for disjunction

Figure 4.2: Modeling Fuzzy Inference using the Fuzzy Petri Net.

Finally, transitions are constructed over these FPN places. The pseudecode of the algorithm is given in Figure 4.3.

The construction of the FPN considers the unification of condition and action calls with event specifications. In this way, we can decide which action execution or condition evaluation generates new events. For these cases, a transition is added from related condition/action FPN places to the generated primitive event in the FPN (*event generating transition*). Our unification principle at FPN construction depends on having the same fuzzy attributes of the same fuzzy domain

```
Begin
while there are still some rules do
  Create a rule object
  Examine Event, Condition, Action parts
  for each one of them do
    Create related FPN places
  end for
end while
Construct fuzzy inference groups
Construct transitions
End
```

Figure 4.3: Construct_FPN Algorithm

in the related FPN places. But they may have different fuzzy linguistic terms.

The rules in the same fuzzy inference group are determined according to their action parts. Rules are in the same fuzzy inference group if action parts unify (their fuzzy attribute domains are the same but they may use different fuzzy linguistic terms). For each fuzzy inference group a combined action and a fuzzy action FPN places are created.

During the FPN construction, the attributes of the rule objects are updated to hold the related links on FPN. These attributes are *PN_primitiveevent*, *PN_fuzzyprimitiveevent*, *PN_fuzzycompositeevent*, *PN_event*, *PN_rule*, *PN _strenghtnedevent*, *PN_fuzzyprimitivecondition*, *PN_fuzzycompositecondition*, *PN_condition*, *PN_clippedaction*, *PN_combineaction* and *PN_fuzzyaction*. In addition, each rule object holds the list of rules in the same fuzzy inference group in its *fuzzyinferencegroup* attribute. Also, each FPN place has a *rule_set* attribute in order to hold the rule objects that uses the FPN place.

Figure 4.4 shows the place types that we use in the FPN. In that figure

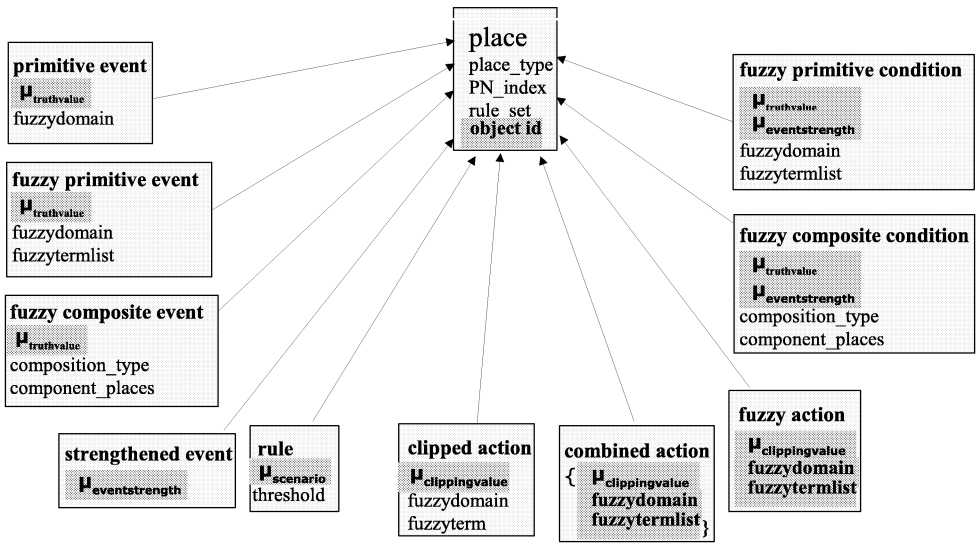


Figure 4.4: Place types used in the FPN

highlighted items are the token variables, whose values are calculated and passed as a parameter during the fuzzy inference.

4.1.3 Example 1 - An Overheating Alarm Generation in a Drive System of 1 Motor (active rule example)

We take this example rules from Bouaziz et al. [16]. The list of the rules in the system:

```
R1:
  ON there is a change in any motor temperature
     AND motor temperature is normal 0.2,
  IF delta is small positive
  THEN alarm notification is low,
R2:
  ON there is a change in any motor temperature
```

```
    AND motor temperature is hot 0.2,
  IF delta is positive
  THEN alarm notification is medium,
R3:
  ON there is a change in any motor temperature
      AND motor temperature is very hot 0.2,
  IF delta is big positive
  THEN alarm notification is high
```

Assume that a typical rule in *emergency* scenario is:

```
Remergency:
           ON there is a change in any motor temperature
              AND motor temperature is very hot,
           IF delta is big positive
           THEN alarm notification is high
```

Suppose that at time t_1 the measured temperature of the motor was 77^oC and it has increased to 132^oC at time t_2.

Steps of the Inference Mechanism

1. *Fuzzify the event*
   $\mu_{normal}(132) = 0.4$, $\mu_{hot}(132) = 0.6$, $\mu_{veryhot}(132)$=0
2. *Calculate the strength of event and check it with the rule threshold*
   Using Equation 3.9 in Section 3.3, following values are calculated for the similarity of rules to the current scenario *emergency*:
   $\mu_{emergency}(R_1)$=0.6, $\mu_{emergency}(R_2)$=0.7, $\mu_{emergency}(R_3)$=1
   and using the formula (1), event strengths are calculated as:
   for R_1 ; 0.6 * $\mu_{normal}(132)$ = 0.6 *0.4 = 0.24
   for R_2 ; 0.7 * $\mu_{hot}(132)$ = 0.7 *0.6 = 0.42
   for R_3 ; 1 * $\mu_{veryhot}(132)$ = 1 *0 = 0
   Threshold values for the rules are given as 0.2, therefore R_1 and R_2 succeed ($0.24 \geq 0.2$ and $0.42 \geq 0.2$ respectively)

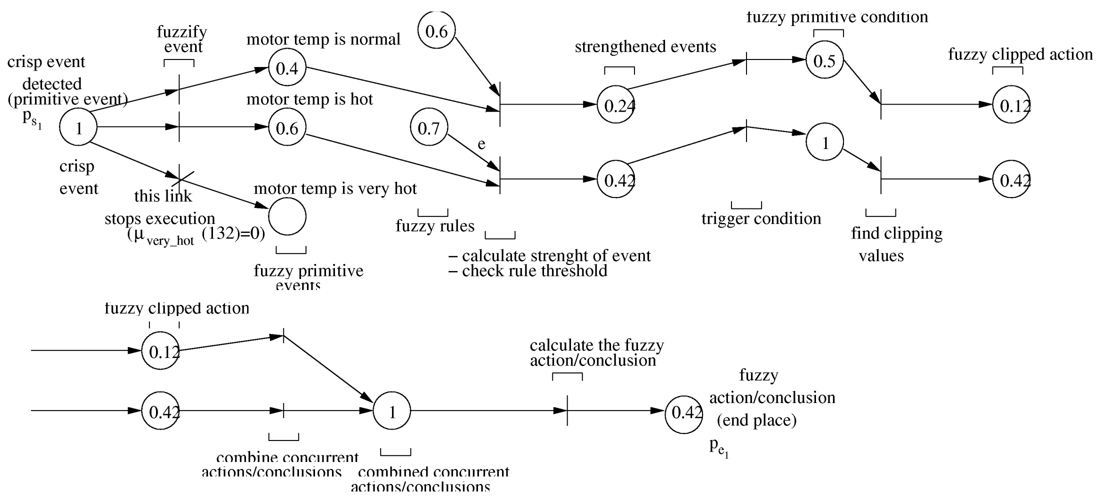

crisp event detected (primitive event)
p_{s_1}
crisp event
fuzzify event
this link stops execution (μ_{very_hot} (132)=0)
motor temp is normal
motor temp is hot
motor temp is very hot
fuzzy primitive events
fuzzy rules
e
– calculate strenght of event
– check rule threshold
strengthened events
fuzzy primitive condition
trigger condition
find clipping values
fuzzy clipped action
fuzzy clipped action
combine concurrent actions/conclusions
combined concurrent actions/conclusions
calculate the fuzzy action/conclusion
fuzzy action/conclusion (end place)
p_{e_1}

Figure 4.5: Modeling alarm treatment in an industrial drive control system by using FPN

3. *Find the condition match factor*

   for R_1; $\mu_{smallpositive}(0.7) = 0.5, min(0.5) = 0.5 \Rightarrow$ match factor $= 0.5$

   for R_2; $\mu_{positive}(0.7) = 1, min(1) = 1 \Rightarrow$ match factor $= 1$

4. *Find the clipping values*

   For this, the condition match factor is multiplied by the strength of event

   for $R_1; 0.24 * 0.5 = 0.12$ (clipping value for the alarm level of `low`)

   for $R_2; 0.42 * 1 = 0.42$ (clipping value for the alarm level of `medium`)

5. *Find the combined fuzzy action*

   We take the maximum of the clipped actions: $max(0.12, 0.42) = 0.42$, which means an alarm level of `medium` is sent.

Figure 4.5 shows how this example problem is solved using FPN.

4.1.4 Example 2 - Automatic Selection of Washing Cycle for a Washing Machine (deductive rule example)

We take this example rules from Yen et al. [78]. List of rules in the system:

```
R1:
  IF Laundry quantity is large
     AND Laundry softness is hard
  THEN washing cycle is strong
R2:
  IF Laundry quantity is medium
     AND Laundry softness is normal hard
  THEN washing cycle is normal
R3:
  IF Laundry quantity is small
     AND Laundry softness is soft
  THEN washing cycle is delicate
R4:
  IF Laundry quantity is medium
     AND Laundry softness is soft
  THEN washing cycle is light
R5:
  IF Laundry quantity is large
     AND Laundry softness is soft
  THEN washing cycle is normal
R6:
  IF Laundry quantity is small
     AND Laundry softness is normal soft
  THEN washing cycle is light
R7:
  IF Laundry quantity is medium
     AND Laundry softness is normal soft
  THEN washing cycle is normal
R8:
  IF Laundry quantity is large
     AND Laundry softness is normal soft
  THEN washing cycle is normal
R9:
  IF Laundry quantity is small
```

```
     AND Laundry softness is normal hard
  THEN washing cycle is light
R10:
  IF Laundry quantity is large
     AND Laundry softness is normal hard
  THEN washing cycle is strong
R11:
  IF Laundry quantity is small
     AND Laundry softness is hard
  THEN washing cycle is light
R12:
  IF Laundry quantity is medium
     AND Laundry softness is hard
  THEN washing cycle is normal
```

The rules related to washing cycle are converted to the following ECA rules internally, whose events are abstract kind.

```
R1:
  ON raise(washing_cycle)
  IF Laundry quantity is large
     AND Laundry softness is hard
  THEN washing cycle is strong
R2:
  ON raise(washing_cycle)
  IF Laundry quantity is medium
     AND Laundry softness is normal hard
  THEN washing cycle is normal


  .... etc
```

Suppose that at time t_1 a query comes like washing_cycle(X).

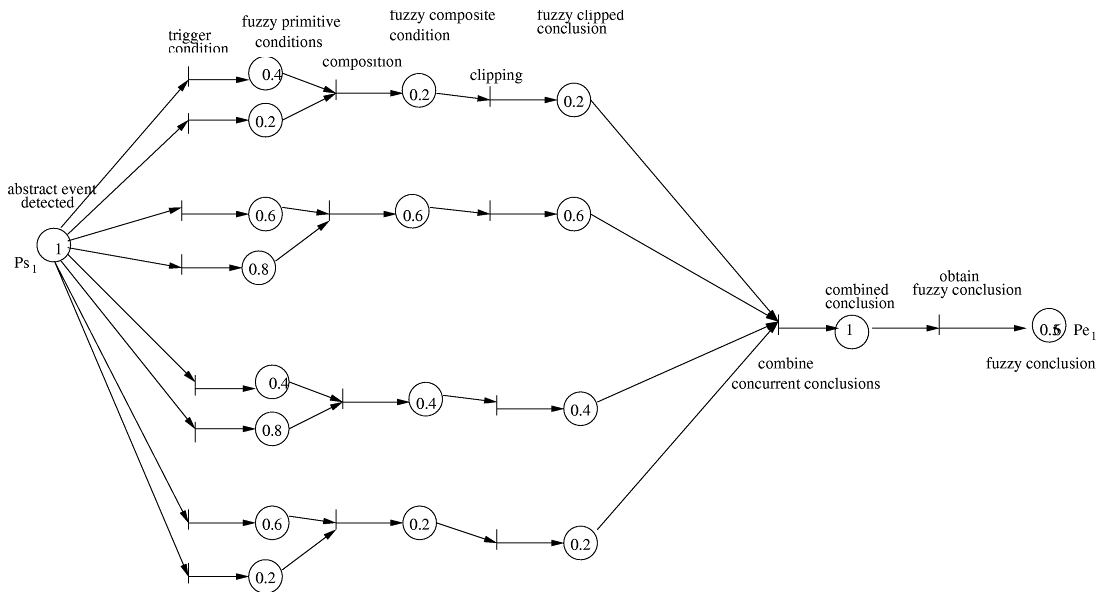


Figure 4.6: Modeling washing cycle selection in a washing machine by using FPN

Steps of Inference Mechanism:

Inference of a deductive rule group starts from evaluation of condition part.

1. *Find the condition match factor*

   for R_1; $\mu_{large}(laundry\_quantity) = 0.4$,

   $\mu_{hard}(laundry\_softness) = 0.2$,

   $min(0.4, 0.2) = 0.2 \Rightarrow$ match factor = 0.2

   for R_2; $\mu_{medium}(laundry\_quantity) = 0.6$,

   $\mu_{normal\_hard}(laundry\_softness) = 0.8$,

   $min(0.6, 0.8) = 0.6 \Rightarrow$ match factor = 0.6

   for R_3; $\mu_{small}(laundry\_quantity) = 0$,

   $\mu_{soft}(laundry\_softness) = 0$,

   $min(0, 0) = 0 \Rightarrow R_3$ fails.

   for R_4; $\mu_{medium}(laundry\_quantity) = 0.5$,

   $\mu_{soft}(laundry\_softness) = 0$,

   $min(0.5, 0) = 0 \Rightarrow R_4$ fails.

   for R_5; $\mu_{large}(laundry\_quantity) = 0.5$,

   $\mu_{soft}(laundry\_softness) = 0$,

   $min(0.5, 0) = 0 \Rightarrow R_5$ fails.

for R_6; $\mu_{small}(laundry\_quantity) = 0$,

$\mu_{normal\_soft}(laundry\_softness) = 0$,

$min(0, 0) = 0 \Rightarrow R_6$ fails.

for R_7; $\mu_{medium}(laundry\_quantity) = 0.6$,

$\mu_{normal\_soft}(laundry\_softness) = 0$,

$min(0.6, 0) = 0 \Rightarrow R_7$ fails.

for R_8; $\mu_{large}(laundry\_quantity) = 0.4$,

$\mu_{normal\_soft}(laundry\_softness) = 0$,

$min(0.4, 0) = 0 \Rightarrow R_8$ fails.

for R_9; $\mu_{small}(laundry\_quantity) = 0$,

$\mu_{normal\_hard}(laundry\_softness) = 0.8$,

$min(0, 0.8) = 0 \Rightarrow R_9$ fails.

for R_{10}; $\mu_{large}(laundry\_quantity) = 0.4$,

$\mu_{normal\_hard}(laundry\_softness) = 0.8$,

$min(0.4, 0.8) = 0.4 \Rightarrow$ match factor $= 0.4$

for R_{11}; $\mu_{small}(laundry\_quantity) = 0$,

$\mu_{hard}(laundry\_softness) = 0.2$,

$min(0, 0.2) = 0 \Rightarrow R_{11}$ fails.

for R_{12}; $\mu_{medium}(laundry\_quantity) = 0.6$,

$\mu_{hard}(laundry\_softness) = 0.2$,

$min(0.6, 0.2) = 0.2 \Rightarrow$ match factor $= 0.2$

2. *Find the clipping values*

   For this, the condition match factor is multiplied by the strength of event.

   For the abstract kind of events $\mu_{event\_strength}$ is always 1.

   for $R_1; 1 * 0.2 = 0.2$ (clipping value for the washing cycle of `strong`)

   for $R_2; 1 * 0.6 = 0.6$ (clipping value for the washing cycle of `normal`)

   for $R_{10}; 1 * 0.4 = 0.4$ (clipping value for the washing cycle of `strong`)

   for $R_2; 1 * 0.2 = 0.2$ (clipping value for the washing cycle of `normal`)

3. *Find the combined fuzzy action*

   We take the maximum of the clipped actions: $max(0.2, 0.6, 0.4, 0.2) = 0.6$, which means washing cycle of `normal` is chosen. Figure 4.6 shows part of the FPN used for this problem.

CHAPTER 5

USING FUZZY PETRI NETS FOR STATIC ANALYSIS OF RULE-BASES

In this Chapter, we check the properties of the system modeled using the FPN. In Section 5.1, we explain the details of how we perform termination analysis on our FPN. Our FPN provides easy checking of this property since it already contains triggering graph information. In Section 5.2, we explain how we guarantee confluence with our fuzzy inference algorithm. For that, we make use of the scenario concept in order to provide a total order for rule executions. We also provide theoretical proofs of our algorithms.

5.1 An Algorithm for Static Analysis on the FPN

Before we check the termination and confluence properties of the system, we construct the FPN, and during the transition construction, we obtain the triggering graph information from the *event generating transitions*. Note that an *event generating transition* is an indication of one rule triggering another.

We use the algorithm given in Figure 5.1 for static analysis on FPN. It first finds the true cycles and informs the user. Once the algorithm detects a cyclic path, it may not be a true cycle due to one or more rules in the cycle having composite events. That is, not all composing events of a composite event may be triggered. These type of false cycles are eliminated in our termination anal-

```
Begin
/*Construct_FPN maps rules to FPN in which
unification is used. The constructed FPN
also holds the triggering graph information.*/
Construct_FPN();

/*determine_true_cycles function calculates
the cyclic paths, it eliminates the false
cycles via considering event compositions*/
if (NULL == (true_cycle_list = determine_true_cycles())) then
    Termination guaranteed;
else
    inform_user(true_cycles_list);
end if

/*check_confluence function controls whether
the rules of the different rule sets
guarantee confluent rule executions*/
if (NULL == (interfering_rule_sets = check_confluence())) then
    Confluent Rule Execution Guaranteed;
else
    inform_user(interfering_rule_sets);
end if
End
```

Figure 5.1: Static Analysis Algorithm

ysis. Next, it checks the interference between the different inference groups to see whether an attribute value written by an inference group is read in another inference group. If this is the case, user is informed about the interfering rule sets.

5.1.1 How to Determine Cycles and True cycles.

For this, we use the following data structures:

a. D is an m x m matrix of rules. Each of its $|i||j|$ entries holds the connectivity information between rules r_i and r_j (or whether there exists an *event generating transition* from r_i to r_j). $D[i][j]$ is 1, if rule r_i is connected to rule r_j with one event generating transition.

b. L is an m x m matrix of rules. Each of its $|i||j|$ entries holds a linked list of rules that are in the path from rules r_i to r_j inclusive.

We calculate the transitive closures of the D matrix. Each of them holds the connectivity information at k edges (or *event generating transitions*) distance. If the i^{th} diagonal at D_k holds a value greater than 0(zero), there is a cycle in k steps (or k *event generating transitions* distance). By the time these D matrices have been calculated, a linked list of rules which have been gone through the path is held in the L matrix. This means that $l_k|i||j|$ holds a linked list of rules which is passed through the path from rules r_i to r_j. If the i^{th} diagonal at D_k holds a value greater than 0(zero), we can obtain the cyclic path elements at $l_k|i||i|$. Now it is easy to find true cycles by checking the $|i||i|$ entry of L_k. If all the rules in the path have only primitive events, this is certainly a true cycle. Otherwise (if any of them has a composite event) the rule set of the primitive events of the rules is examined. If they are all included in the cycle, again the cycle is a true cycle. Otherwise it is not a true cycle. If there is at least one true cycle, the user is informed about the situation together with the rules taking place inside the cycle. Then it is the user's choice to change the definitions of these rules. After that, the termination analysis is repeated.

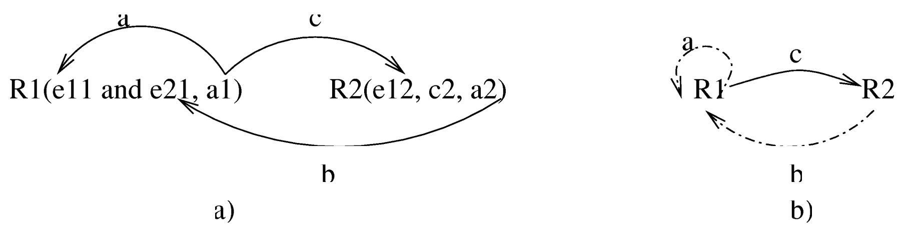


Figure 5.2: Dependencies and the Triggering Graph for the Example

5.1.2 Example

Suppose there are the following two rules:

```
R1:
  ON (e11 and e21)
  IF c1
  THEN a1
R2:
  ON e12
  IF c2
  THEN a2
```

In these rules a_1 unifies with crisp event e_1 (which has e_{11} and e_{12} as its fuzzy primitive events) and a_2 unifies with crisp event e_2 (which has e_{21} as its fuzzy primitive event). The dependencies and the triggering graph are shown in Figure 5.2. Dashed arcs show the partial triggering due to composite events and solid arcs show the total triggering. Figure 5.3 shows the FPN constructed according to this rule set.

For this example, D_1 and L_1 are as follows:

$$D_1 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} L_1 = \begin{bmatrix} 1,1 & 1,2 \\ 2,1 & 0 \end{bmatrix}$$

Since $D_1|1||1|$ entry has the value 1, this means that there is a cycle. After checking $r_1.PN\_primitiveevent.rule\_set$, rules r_1 and $r2$ are found. Since the cyclic set $\{1, 1\}$ does not contain $r2$, this is not a true cycle. Therefore it is excluded. Then D_2 is calculated and at the same time L_2 is obtained:

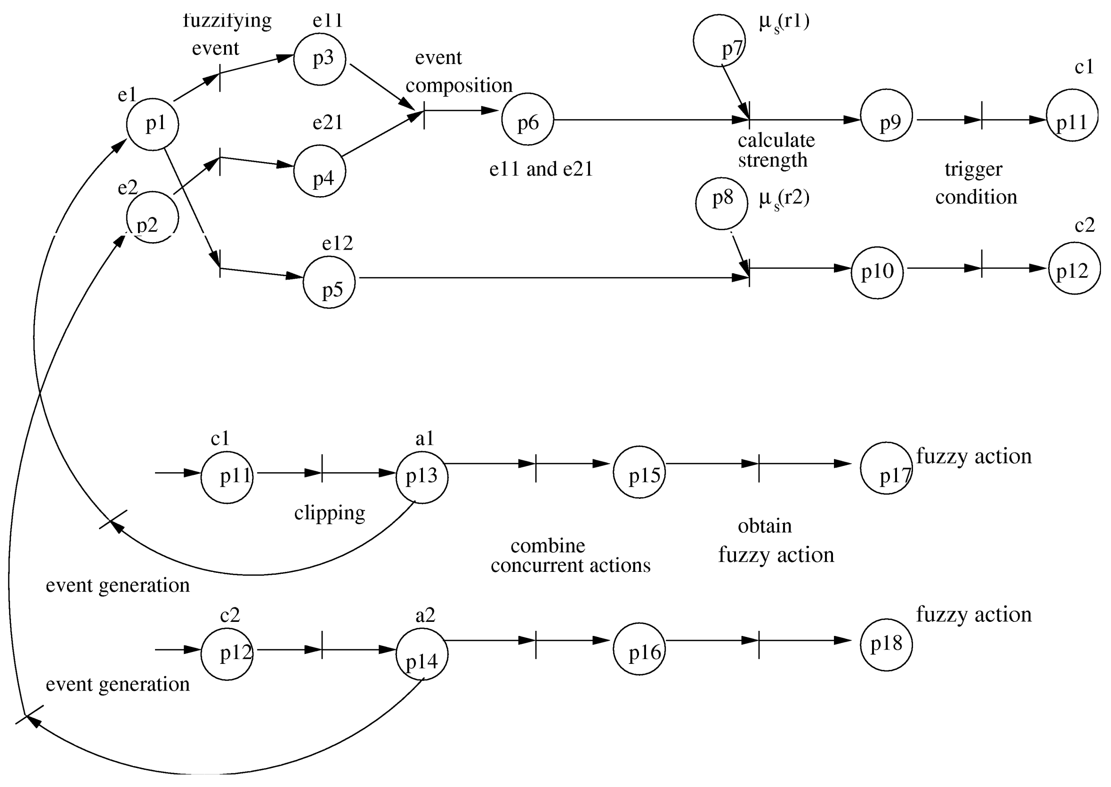


Figure 5.3: FPN constructed for the Example

$$D_2 = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} L_2 = \begin{bmatrix} 1,1,1;1,2,1 & 1,1,2 \\ 2,1,1 & 2,1,2 \end{bmatrix}$$

Since $D_2|1||1|$ entry has a value greater than 0(zero), there is a possibility of a cycle. $r_1.PN\_primitiveevent.rule\_set$ is the rules r_1 and r_2. Since the cyclic set $\{1,2,1\}$ contains both r_1 and r_2, it is a true cycle. The user is informed about the cyclic situation together with the rules taking place in the cycle. The rules inside the cycle are r_1 and r_2. So, termination analysis returns the rules r_1 and r_2. This means that they are part of the cycle and need to be changed in order to break the cycle.

5.1.3 The Algorithm for Determining the True Cycles

Lemma 5.1.3.1: *Let there be k rules in the rule base. The algorithm determining the true cycle decides on termination in at most k steps.*

Proof: If we have k rules in the rule base, we can have at most k distinct edges

```
Begin
calculate $L_1$
Initialize k to 1
while ((D_k ≠ 0) and (k ≤ number of rules)) do
  for each rule $r_i$ do
    //i is the index of the rule $r_i$ in the D matrix
    if $D_k|i||i| \geq 1$ then
      if rules in the $L_k|i||i|$ are distinct then
        if none of the rules contains composite event then
          return cyclic rules ($L_k|i||i|$)
        else
          if all rule sets of the primitive events of the rules (which have
          composite event) are included in the $L_k|i||i|$ then
            return cyclic rules ($L_k|i||i|$)
          end if
        end if
      end if
    end if
  end for
  Increment k by 1
  calculate $D_k$
  calculate $L_k$
end while
return no cycle
End
```

Figure 5.4: Determine_true_cycles Algorithm

to go through if there is cyclicity. This is due to the fact that if there are k nodes, there may be at most k edges connecting these nodes in a circular path (considering that the nodes $n_1, n_2, ..., n_{(k-1)}, n_k, n_1$ can be gone through and this path has k edges)

Lemma 5.1.3.2: *Let there be k rules in the rule base and all rules have primitive events. If there is no cyclic behavior, D_n matrix becomes 0(zero) at finite number of steps which is at most k.*
Proof: If we have k rules in the rule base and at the same time if all rules have primitive events, there can be at most $k-1$ distinct edges to go through if there is no cyclicity. This is due to the fact that if there are k nodes, there may be at most $k-1$ edges connecting these nodes (considering that the nodes $n_1, n_2, ..., n_{(k-1)}, n_k$ can be gone through and this path has $k-1$ edges). In these cases k^{th} edge goes nowhere and the D_n matrix becomes 0(zero).

Lemma 5.1.3.3: *Let rule r_i have a composite event. Although the $D_n|i||i|$ entry is not* 0*(zero) at some steps, it does not guarantee non-termination.*
Proof: If r_i has a composite event, although the $D_n|i||i|$ entry is not 0(zero) (there is a cyclic triggering of the r_i), all the rules related to triggering the events of the r_i may not be taking place in the cyclic path. If this is the case, there is actually no true cycle.

5.2 Guaranteeing Confluent Executions in Our FPN Model

After the FPN gets constructed, the system checks whether the defined rules naturally guarantees confluent rule executions or not by controlling the interference of rule sets in different inference groups (Algorithm given in Figure 5.5).

If a rule set reads in its event part or condition part whatever is written in action part of another rule set, this means that the system may not provide confluent executions. In those cases, we put another mechanism for confluent executions, which is explained below.

Before explaining how to guarantee confluent executions on FPN, we shall give

```
Input: Inference groups on FPN
Output: Interfering inference groups $I_R$
Begin
for (each inference group $inf_i$) do
  for (each inference group $inf_j$) do
    if ($read\_set(inf_i) == write\_set(inf_j)$) then
      add $inf_i$ and $inf_j$ to $I_R$
    end if
  end for
end for
return $I_R$
End
```

Figure 5.5: Check_Confluence Algorithm

the pseudecode of our fuzzy inference algorithm based on the FPN in Figure 5.6. In that, we use some data structures for the internal representation of the FPN.

a. M is an m x 1 column vector of places p_i, where i=1, ..., m. Each of its entries holds a data structure of three elements:

   - 1st element holds the place id.

   - 2nd element holds the object id which uses the place,

   - 3rd element holds the token value.

b. *current_places* is a linked list. Each of its entries holds a data structure of two elements:

   - 1st element holds the place id.

   - 2nd element holds the index of place id in M vector.

```
Begin
find the index of the start place $P_{si}$ in $M$, put a token to there and mark it
if the rules triggered are in the same inference group then
  set the active inference group to the inference group of the rules
else
  order the triggered rules according to their scenario similarity
  set the linked list of active inference groups according to this order
end if
while linked list of active inference groups is not consumed yet do
  initialize the current places list with primitive events of active inference group
  while there are some current places do
    if head of current places list cpi is used by active inference group then
      for each of the transitions where current place cpi is an input do
        if the rule(s) using the transition is in active inference group then
          if other input places of the transition also have tokens then
            fire the transition, mark the output places in $M$ using $c_{ij}^{+}$
            add the marked output place to the tail of the current places list
          end if
        end if
      end for
    end if
    if the current place cpi is an end place then
      add the cpi to the reached end places list $P_e$
    end if
    remove the current place cpi from the current places list
  end while
  update the head of the linked list of active inference groups to the next
end while
return $P_e$
End
```

Figure 5.6: Inference Algorithm

c. N is an n x 1 column vector of transitions t_j, where j=1, ..., n. Each of its entries holds a transition function. Each transition function uses the elements of the input places *token_values* and produces an output for the output places *token_values*.

d. C^+=(c^+_{ij}) and C^-=(c^-_{ij}) represent the output incidence matrix and input incidence matrix respectively, where c^+_{ij} is 1 if there is an arc from the transition t_j to place p_i and c^-_{ij} is 1 if there is an arc from the place p_i to the transition t_j and their values are 0(zero) if there is no connection.

Lemma 5.2.1: *The inference algorithm based on FPN guarantees confluent rule execution via providing a total order between different rule sets.*

Proof: In the inference algorithm that we give above, if the rules triggered are in the same fuzzy inference group, their execution is carried out at the same time and the total outcome is computed. On the other hand, if the rules triggered are not in the same fuzzy inference group, total ordering is achieved (i.e. rules are executed in an order) according to their $\mu_s(r)$ (the similarity to the current scenario) values. Therefore, the confluent rule execution is guaranteed.

CHAPTER 6

IMPLEMENTATION

6.1 Architecture of the system using FPN based inference mechanism

Figure 6.1 shows the architecture of the system using FPN based inference mechanism. In that:

User Interface receives the rule definitions and query requests from the user. It provides user the output of the queries or information/alarm actions due to a dynamic occurence.

Rule Controller is responsible for the storage and retrieval of the event/rule definitions. It also provides the construction of FPN after all the rules get defined. In addition, each time a query comes or an event occurs, the rule controller triggers the Inference Engine for execution.

Primitive Event Detector is responsible from the detection of primitive events, and informs the Rule Controller about the occurrence of primitive events. We consider some attribute changes made by user, which affect the active rule evaluation, as if they come from the sensor.

Fuzzy Petri Net Based Inference Engine collects primitive event occurences after which it decides the right time for the execution on the FPN. After the execution on FPN, it sends the output to the user interface.

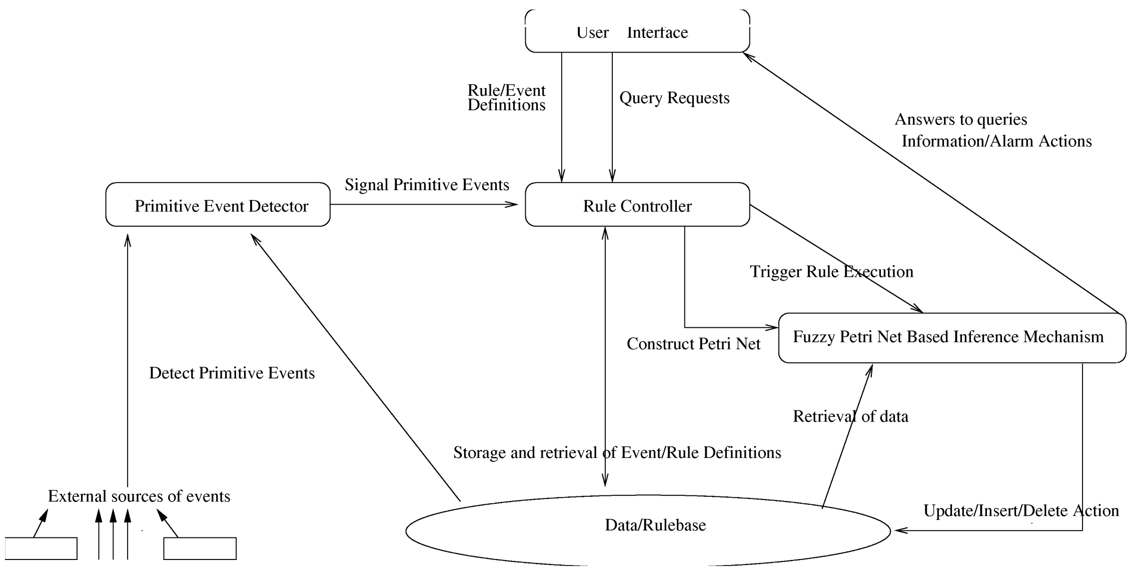


Figure 6.1: Architecture of the System

6.2 Implementation Platform

We implemented a prototype system of the *Intelligent Database* architecture. We used ITASCA for the implementation of the object-oriented database and Borland C++ Builder for the implementation of the user interface and application servers, which are FPN constructor, fuzzy processor and inference engine. The implementation platform is shown in Figure 6.2.

The main reason for using ITASCA Object Database Management System was its availability. In addition, it supports our main requirements for an object oriented database management system. ITASCA Object Database Management System is integrated into the application via C++ API library.

Borland C++ is used for the implementation of the graphical user interface of the system. User is able to create, modify and delete fuzzy types, similarity matrices, membership functions, classes, objects, rules and scenarios through the user interface. The FPN constructor and inference engine is also implemented by using Borland C++.

A front-end processor is developed to provide the fuzzy processor functionality. It maps requests from the user interface to the appropriate format for the object-oriented database or vice versa. It also handles the requests of FPN based inference engine from the object-oriented database.

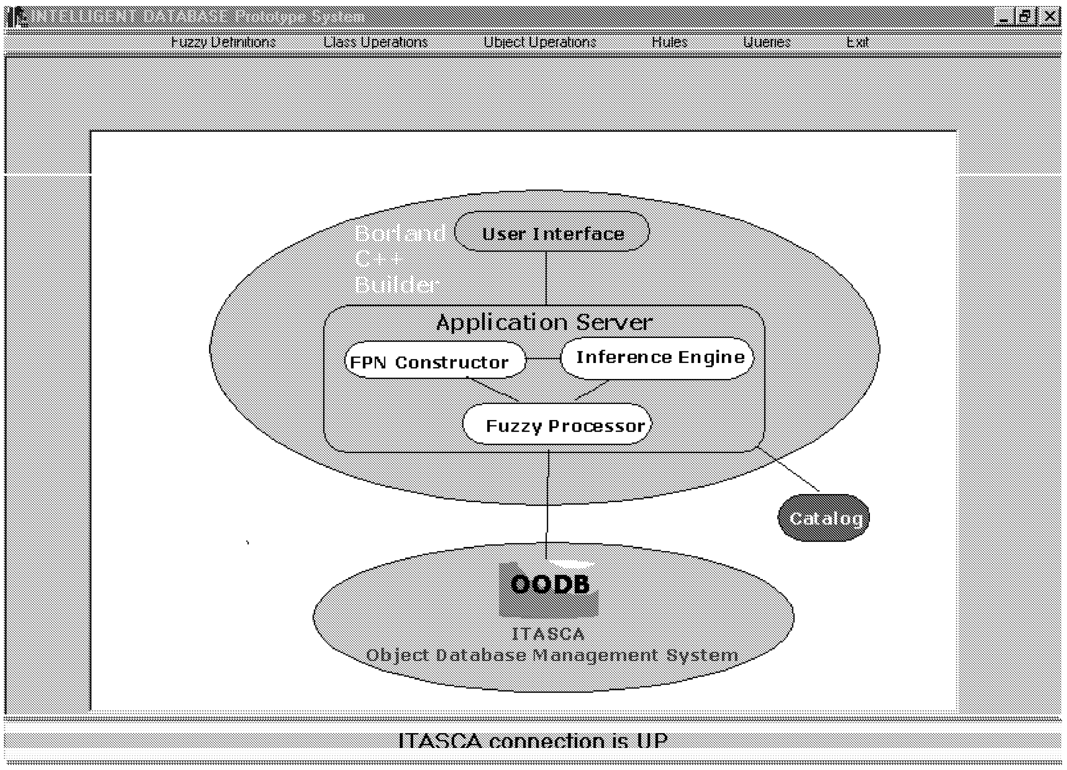


Figure 6.2: The implementation platform of the prototype system

A catalog is held to keep necessary information as a short look up table about the application. The information stored in the catalog is about the classes, fuzzy attributes, derived attributes, rules, scenarios and some information related to the constructed FPN. The inference on the FPN runs according to the information stored in the catalog.

6.3 Module Structure of the Prototype System

We have 5(five) modules in our prototype system. The interaction between these modules are given in Figure 6.3.

6.3.1 Fuzzy Definitions

In this module, user defines fuzzy types, similarity matrices and membership functions. Fuzzy type of a domain should be defined before similarity matrices and membership functions of the domain. A fuzzy type definition of a domain should include:

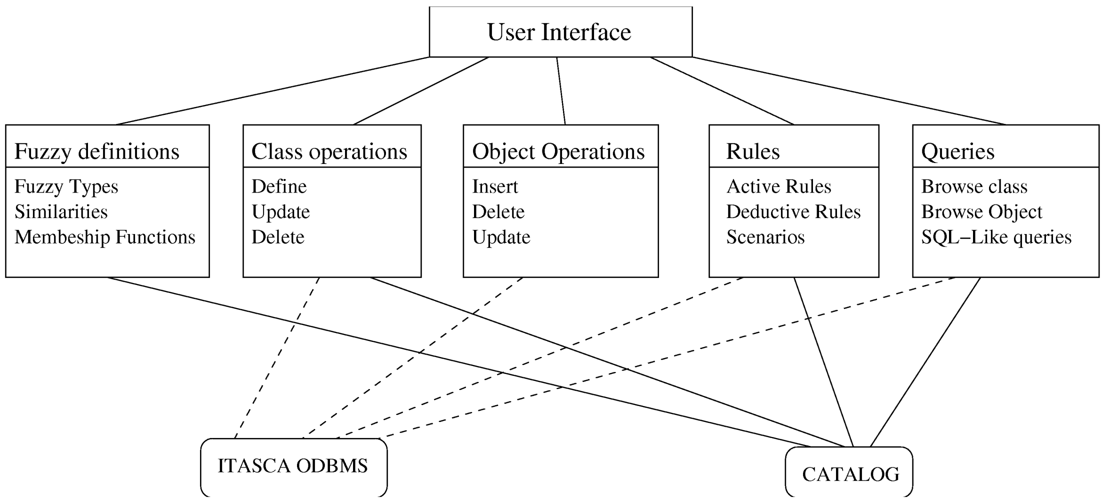


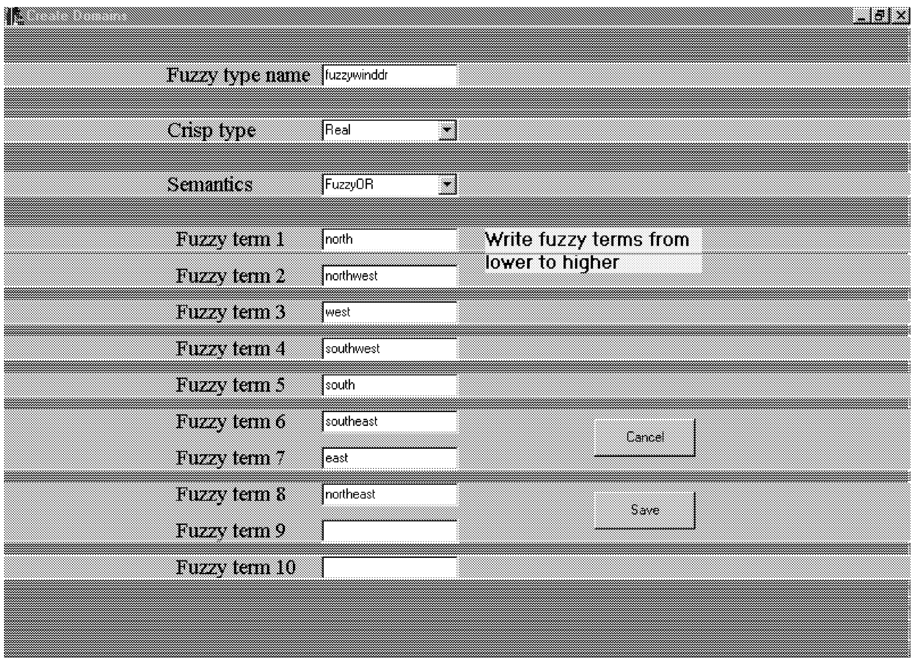
Figure 6.3: Module Structure of the prototype system

- fuzzy type name,
- crisp type,
- semantic, and
- fuzzy terms

For the example, direction of a wind can be defined as a fuzzy type. Let its fuzzy type name be *fuzzywinddr*, its crisp type be *real*. It has a *fuzzyOR* semantic, and it contains the fuzzy terms *north, northwest, west, southwest, south, southeast, east* and *northeast*. An object attribute defined with this type will have either a real value or a set of fuzzy values specified in the definition. Figure 6.4 shows the screen used for fuzzy type definitions.

Since our application domain require a sensor to be connected to feed the system with input data, we think that those object attributes will have crisp values. However, our rules all include fuzzy terms. Therefore, each attribute is defined with a fuzzy type, and fuzzified before being used by a rule.

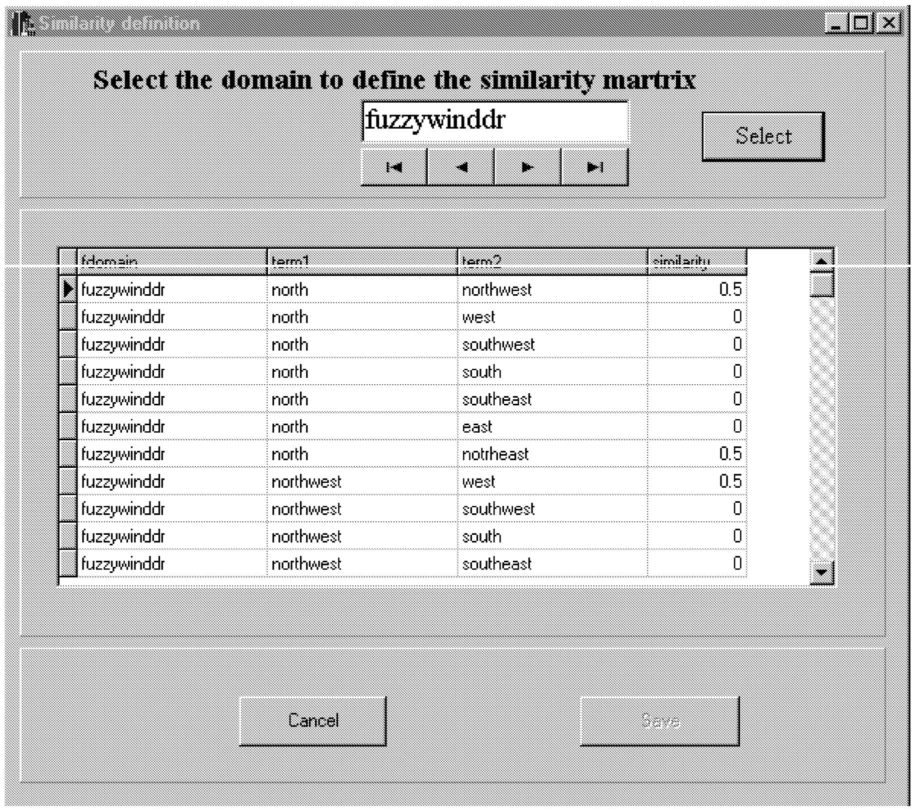
In the similarity matrix definition phase, it is asked to define similarities for all pairs of attributes. Reflexivity and symmetry properties of similarity matrices are considered during the similarity definitions. Figure 6.5 shows the screen used for similarity definitions.



Fuzzy type name	fuzzywinddr
Crisp type	Real
Semantics	FuzzyOR
Fuzzy term 1	north
Fuzzy term 2	northwest
Fuzzy term 3	west
Fuzzy term 4	southwest
Fuzzy term 5	south
Fuzzy term 6	southeast
Fuzzy term 7	east
Fuzzy term 8	northeast
Fuzzy term 9	
Fuzzy term 10	

Write fuzzy terms from lower to higher

Figure 6.4: Fuzzy type definition

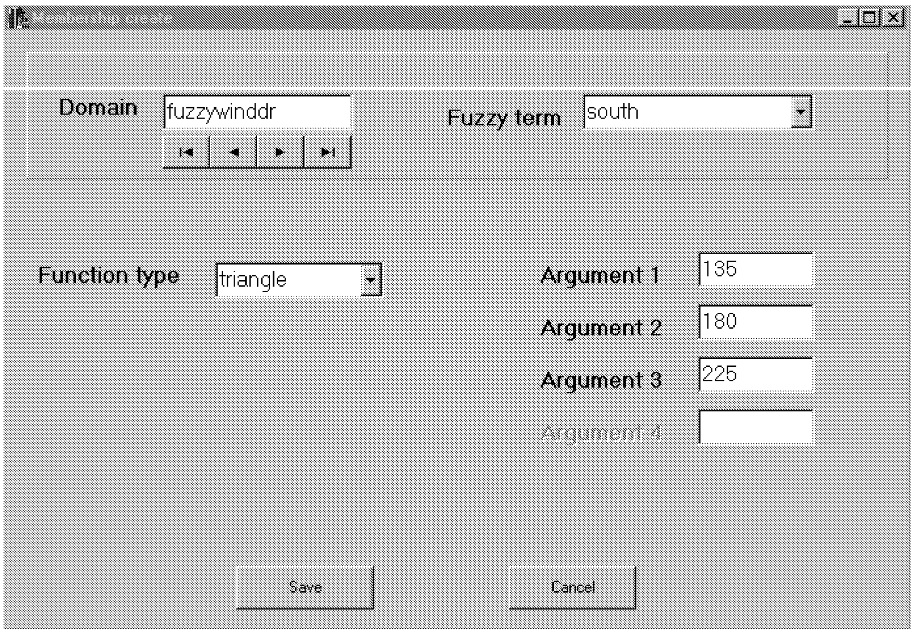


Select the domain to define the similarity martrix

fuzzywinddr

fdomain	term1	term2	similarity
fuzzywinddr	north	northwest	0.5
fuzzywinddr	north	west	0
fuzzywinddr	north	southwest	0
fuzzywinddr	north	south	0
fuzzywinddr	north	southeast	0
fuzzywinddr	north	east	0
fuzzywinddr	north	notrheast	0.5
fuzzywinddr	northwest	west	0.5
fuzzywinddr	northwest	southwest	0
fuzzywinddr	northwest	south	0
fuzzywinddr	northwest	southeast	0

Figure 6.5: Similarity matrix definition



Domain	fuzzywinddr
Fuzzy term	south
Function type	triangle
Argument 1	135
Argument 2	180
Argument 3	225
Argument 4	

Figure 6.6: Membership function definition

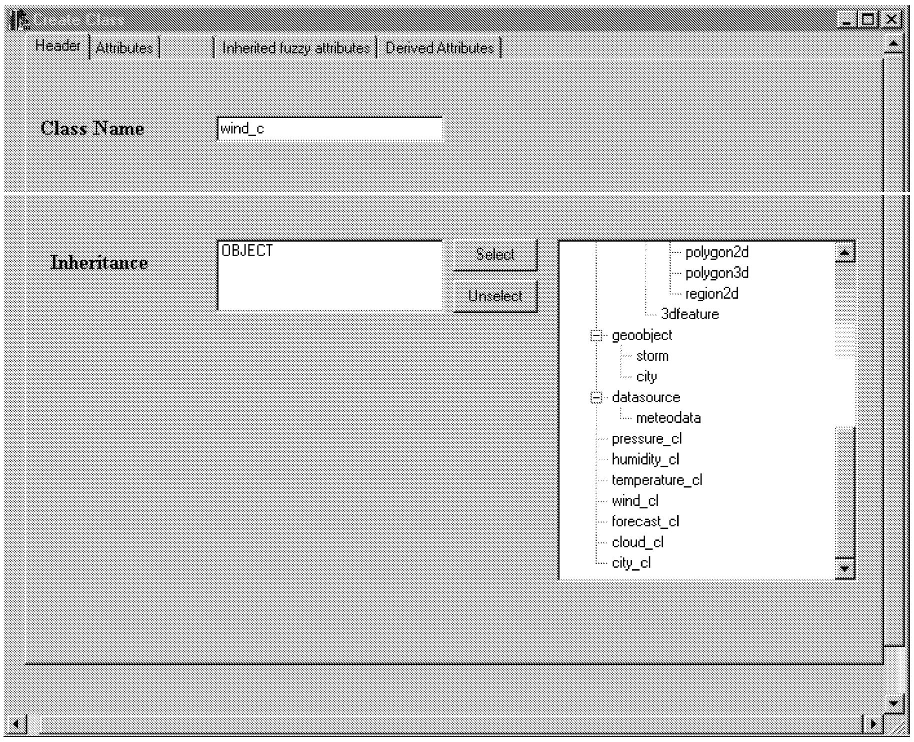
Membership functions are defined by selecting a fuzzy type. The system then brings the fuzzy terms for their membership function definitions. After the user chooses a suitable membership function, such as trapezoidal, triangular, etc., he/she provides required function arguments. Figure 6.6 shows the screen used for membership definitions.

It is also allowed user to update or delete fuzziness related definitions. All these definitions are stored in the system catalog.

6.3.2 Class Operations

Classes should be defined after the fuzzy type definitions. Otherwise fuzzy types can't be visible during the class definitions. A class can be defined with the following information from the user:

- class name,
- superclasses,
- attribute definitions, and
- derived attribute definitions.

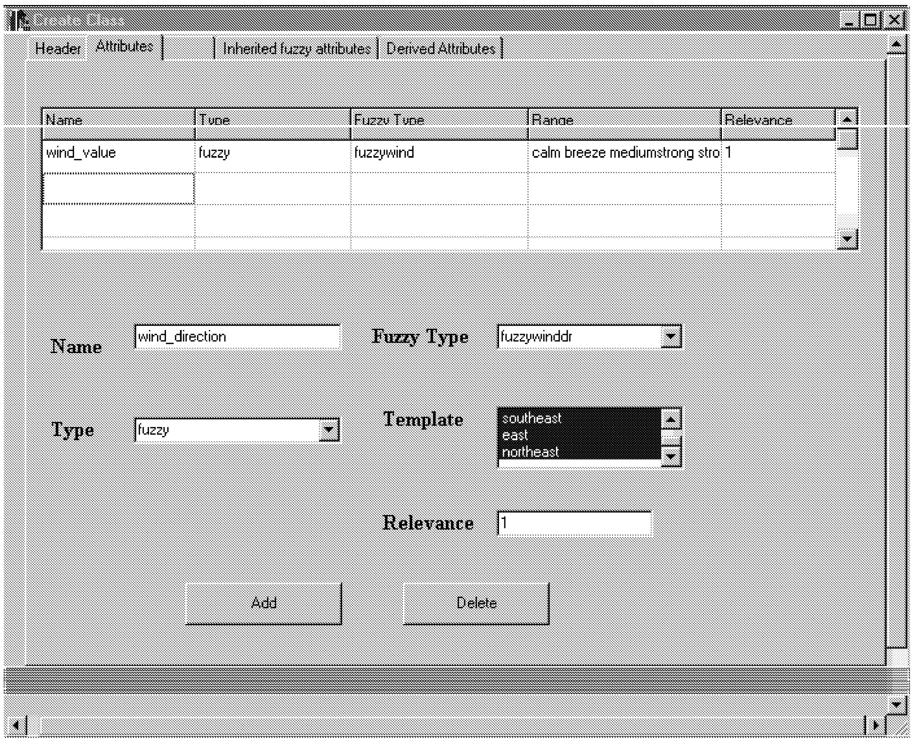


Class Name: wind_c

Inheritance: OBJECT

polygon2d
polygon3d
region2d
3dfeature
geoobject
storm
city
datasource
meteodata
pressure_cl
humidity_cl
temperature_cl
wind_cl
forecast_cl
cloud_cl
city_cl

Figure 6.7: Class creation (1)



Name	Type	Fuzzy Type	Range	Relevance
wind_value	fuzzy	fuzzywind	calm breeze mediumstrong stro	1

Name: wind_direction

Fuzzy Type: fuzzywinddr

Type: fuzzy

Template: southeast, east, northeast

Relevance: 1

Figure 6.8: Attribute Definition - Class Creation(2)

Once a class is defined (Figures 6.7 to 6.9), its definition is inserted into ITASCA object database system. Also some required information is stored in the application catalog. User can delete class definitions.

6.3.2.1 Derived Attributes

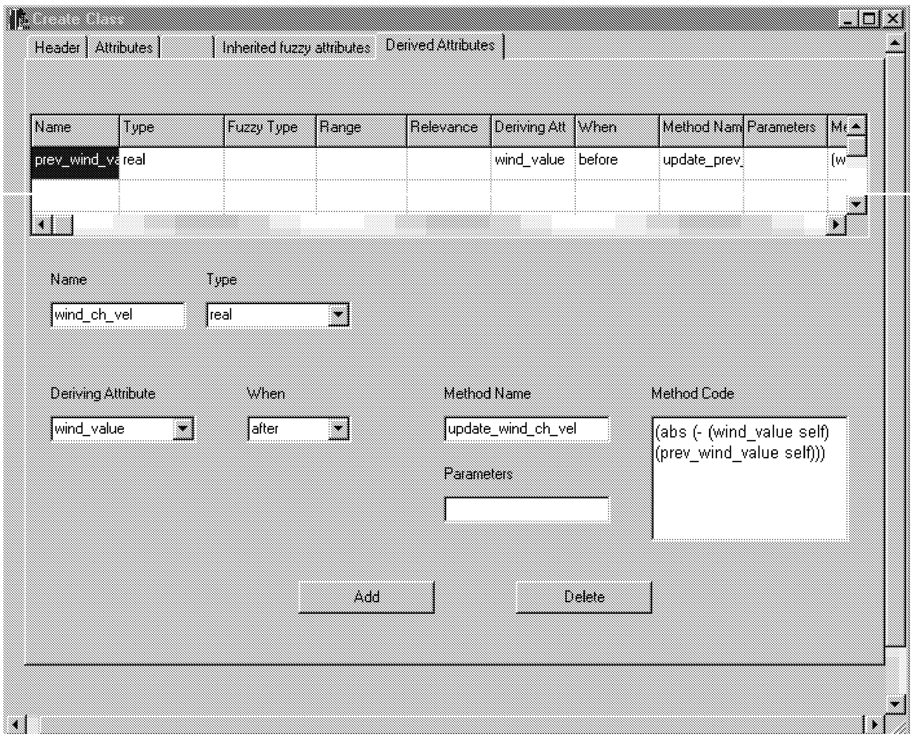
Derived attributes are defined during the class definition. First, user should define simple attributes (i.e. not the derived attributes) as shown in Figure 6.8. Then user can define derived attributes from these attributes. Definition of a derived attribute includes:

- attribute name for the deriving attribute
- attribute name for the derived attribute
- when to derive (either before or after)
- method name
- method code

Derived attributes are updated automatically when there is an update to the deriving attribute. If the derived attribute uses the previous value of the deriving attribute, *when* should be *before*, and attribute gets derived just before the update on deriving attribute. If the derived attribute needs the new value of the deriving attribute then *when* should be *after*. An ITASCA method is used for the derivation. Method code is a zero or more lisp forms, which is defined by the user. This is the only way of defining methods in ITASCA at run_time. In the method code, *self* is the variable referencing the object to which the method will be applied. Derived attributes can also be of fuzzy type.

Derived attributes are stored as any attribute in ITASCA class schema. However, we don't allow them to be updated by the user. Their update is controlled by the system. We keep information related to the derived attributes in the catalog. This information is used by the system for their update.

Consider the weather forecasting application given in Chapter 7. In that, wind should have attributes like *wind_value*, *prev_wind_value* and *wind_ch_vel* etc. In


Header | Attributes | | Inherited fuzzy attributes | Derived Attributes

Name	Type	Fuzzy Type	Range	Relevance	Deriving Att	When	Method Nam	Parameters	Me
prev_wind_va	real				wind_value	before	update_prev_		(w

Name: wind_ch_vel
Type: real
Deriving Attribute: wind_value
When: after
Method Name: update_wind_ch_vel
Parameters:
Method Code:

```
(abs (- (wind_value self)
(prev_wind_value self)))
```

Figure 6.9: Derived Attribute Definition - Class Creation(3)

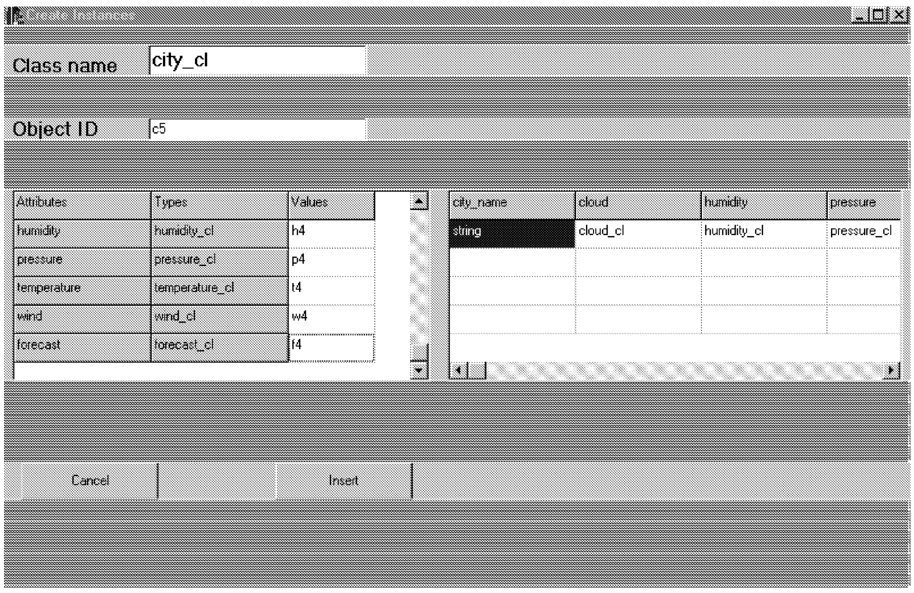
this example, *prev_wind_value* is of a *before* type derived attribute, which should be updated before the *wind_value* gets updated. However, *wind_ch_vel* is of an *after* type derived attribute, which should be updated after the *wind_value* gets updated. Figure 6.9 shows how the derived attributes of wind gets defined.

6.3.3 Object Operations

User can create, update and delete objects of any class. An object is defined with:

- object id,
- class name, and
- attribute values which may be of fuzzy type.

During the object creation, once the user selects a class, all the attributes of the class are listed for their value enterance. Note that derived attributes of the class are not visible at this point. Since objects are stored in the database, this



Class name: city_cl

Object ID: c5

Attributes	Types	Values
humidity	humidity_cl	h4
pressure	pressure_cl	p4
temperature	temperature_cl	t4
wind	wind_cl	w4
forecast	forecast_cl	f4

city_name	cloud	humidity	pressure
string	cloud_cl	humidity_cl	pressure_cl

Figure 6.10: Object creation

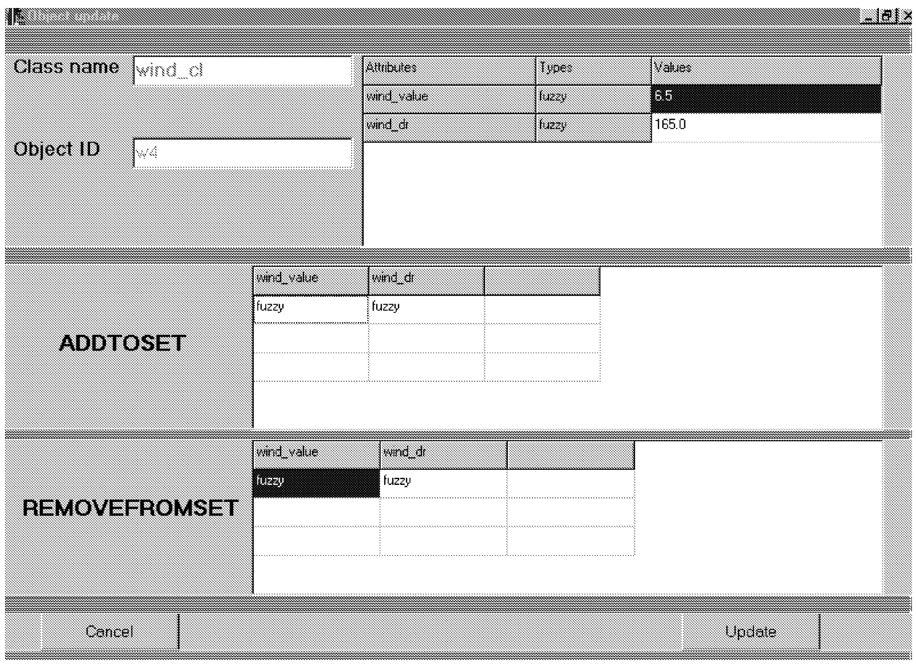
operation is related with ITASCA object database system. Figures 6.10 and 6.11 show the parts of the user interface dealing with objects.

6.3.4 Rules

Users are allowed to define, active, deductive and scenario rules (Figures 6.12 to 6.15). However, all the rules are mapped to an active type internally. Rule definitions include the class name that it is applied, rule id, event (if an active or a scenario rule), condition, action parts and an event threshold (if active rule). Once a rule object is defined, its related FPN places are automatically created, and related links to these FPN places are set in the attributes of the rule. Rule definitions are stored in catalog. Information for constructing the FPN and providing a fuzzy inference mechanism on that is also stored in the catalog.

After the rule definitions, system automatically performs static analysis of the rule base. In that termination and confluence properties are checked. Also, Petri Net constructed by the system can be viewed at any time. Related screens for these are shown in Chapter 7 at Figures 7.7, 7.8, and Figures 7.9 to 7.15 for the example rules of the weather forecasting application.

Once the scenario rules are defined current scenario can be updated at any


Object update

Class name wind_cl

Object ID w4

Attributes	Types	Values
wind_value	fuzzy	6.5
wind_dr	fuzzy	165.0

ADDTOSET

wind_value	wind_dr	
fuzzy	fuzzy	

REMOVEFROMSET

wind_value	wind_dr	
fuzzy	fuzzy	

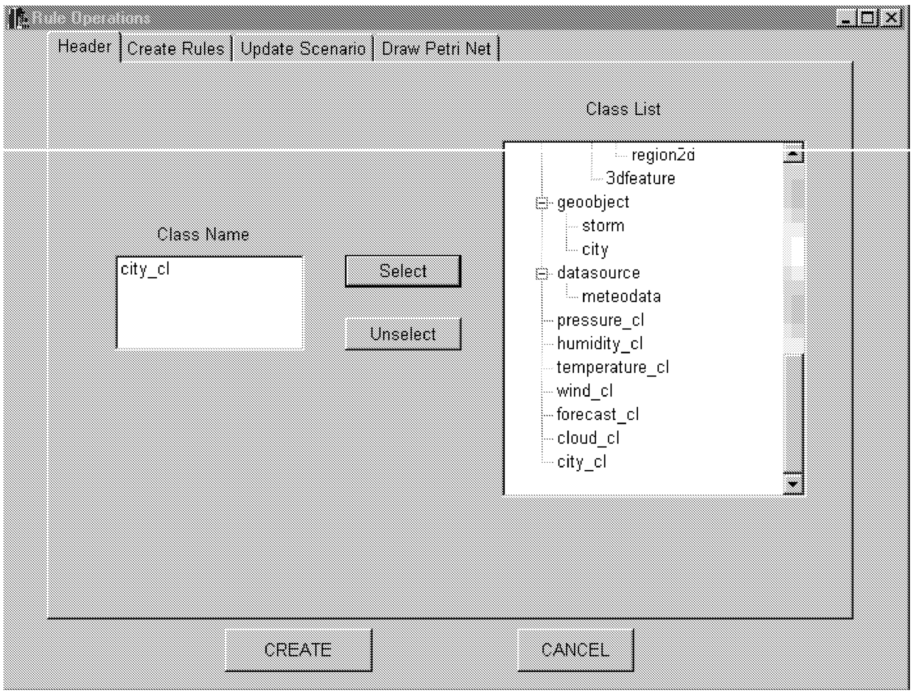
Figure 6.11: Object update

time (Figure 6.16). On each scenario update, similarities of active rules to the selected scenario is calculated and stored in catalog automatically.

6.3.5 Queries

The system provides three different query facilities: class browser, object browser and SQL-like queries. User can access the objects of any class using the class browser. If specific objects are requested, object browser should be used with the filters to set criteria to select specific objects. Screens of class and object browsers can be seen in Figures 6.17 and 6.18.

SQL-like queries are of select-from-where type. Users can formulate queries with fuzzy predicates. This type of queries can be related to the data stored in ITASCA or deductive rules. If the query is related to attributes (either derived or not) stored in ITASCA, simply the objects satisfying the query are filtered. However, if the query requires the fuzzy inference of a deductive rule, it is assumed as if an abstract event is generated. Then inference is performed on the FPN, whose result then becomes the result of the SQL-like query. For example, if the *hum_value* of the *humidity* attribute of a *city* object is inquired, it is a simple

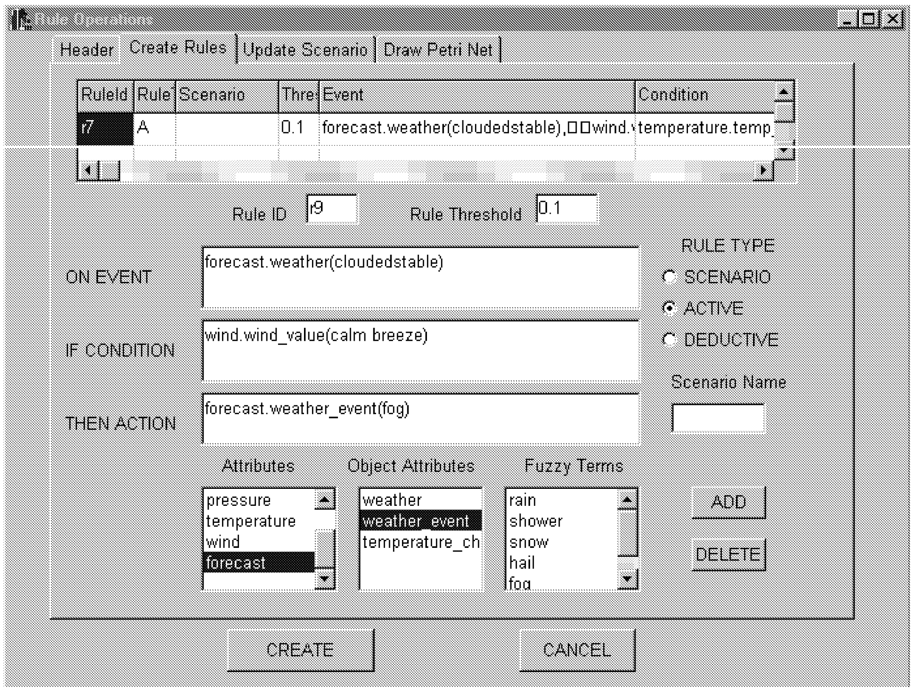


Class List

- region2d
- 3dfeature
- geoobject
  - storm
  - city
- datasource
  - meteodata
- pressure_cl
- humidity_cl
- temperature_cl
- wind_cl
- forecast_cl
- cloud_cl
- city_cl

Class Name: city_cl

Figure 6.12: Rule operations (1)



RuleId	Rule	Scenario	Thre	Event	Condition
r7	A		0.1	forecast.weather(cloudedstable),□□wind.\	temperature.temp_

Rule ID: r9 Rule Threshold: 0.1

ON EVENT: forecast.weather(cloudedstable)

IF CONDITION: wind.wind_value(calm breeze)

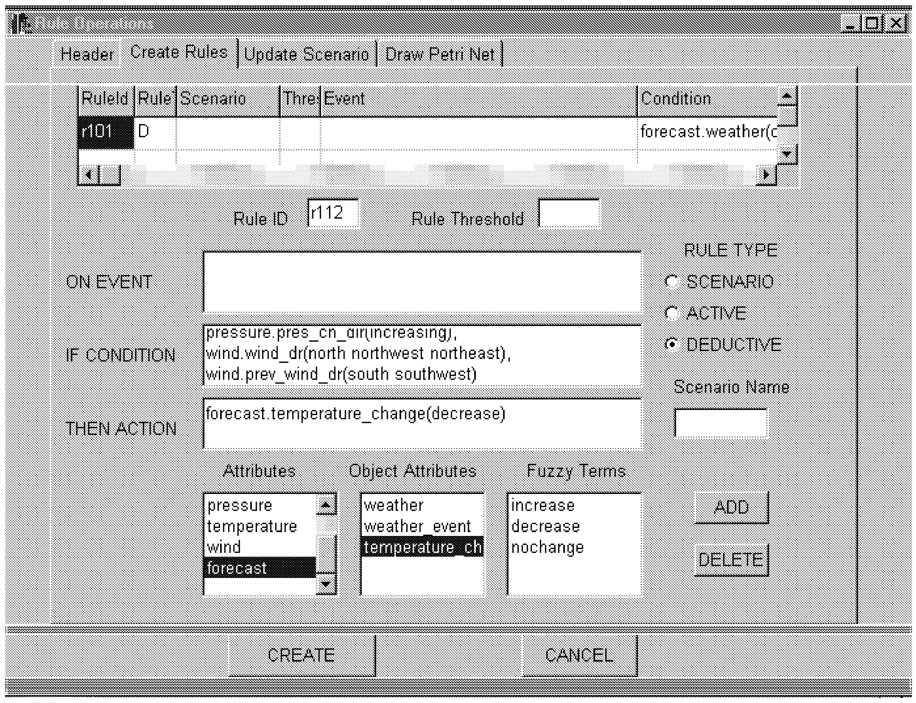
THEN ACTION: forecast.weather_event(fog)

RULE TYPE: SCENARIO [✓ ACTIVE] DEDUCTIVE

Scenario Name:

Attributes	Object Attributes	Fuzzy Terms
pressure	weather	rain
temperature	weather_event	shower
wind	temperature_ch	snow
forecast		hail
		fog

Figure 6.13: Active Rule Definition - Rule operations (2)



RuleId	Rule	Scenario	Thre	Event	Condition
r101	D				forecast.weather(c

Rule ID: r112
Rule Threshold:

ON EVENT:

IF CONDITION:
```
pressure.pres_cn_dir(increasing),
wind.wind_dr(north northwest northeast),
wind.prev_wind_dr(south southwest)
```

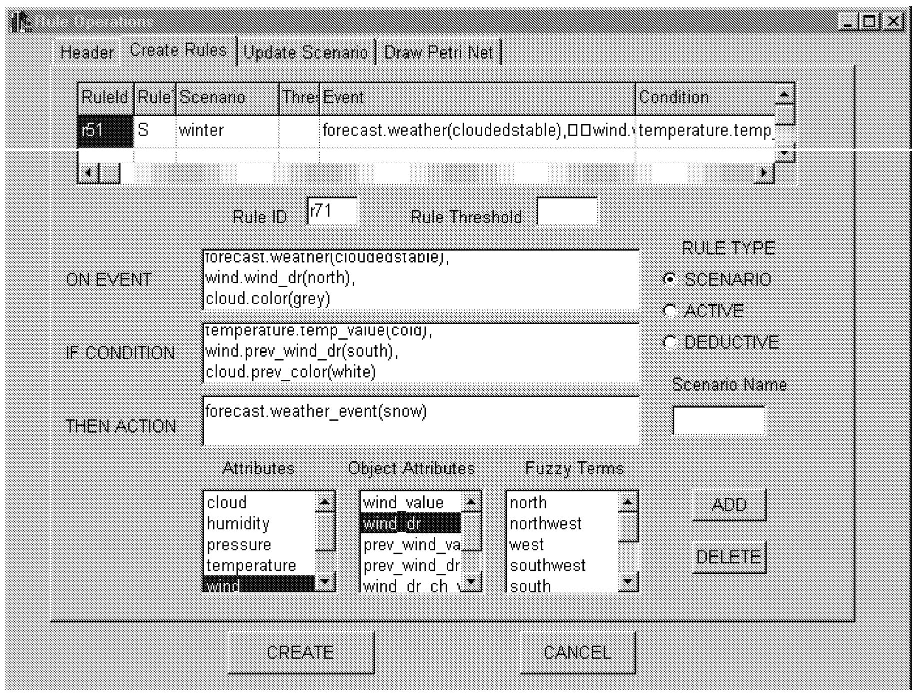
THEN ACTION:
```
forecast.temperature_change(decrease)
```

RULE TYPE: ( ) SCENARIO ( ) ACTIVE (● DEDUCTIVE)

Scenario Name:

Attributes	Object Attributes	Fuzzy Terms
pressure	weather	increase
temperature	weather_event	decrease
wind	temperature_ch	nochange
forecast		

Figure 6.14: Deductive Rule Definition - Rule operations (3)



RuleId	Rule	Scenario	Thre	Event	Condition
r51	S	winter		forecast.weather(cloudedstable),□□wind.\	temperature.temp_

Rule ID: r71
Rule Threshold:

ON EVENT:
```
forecast.weather(cloudedstable),
wind.wind_dr(north),
cloud.color(grey)
```

IF CONDITION:
```
temperature.temp_value(cold),
wind.prev_wind_dr(south),
cloud.prev_color(white)
```

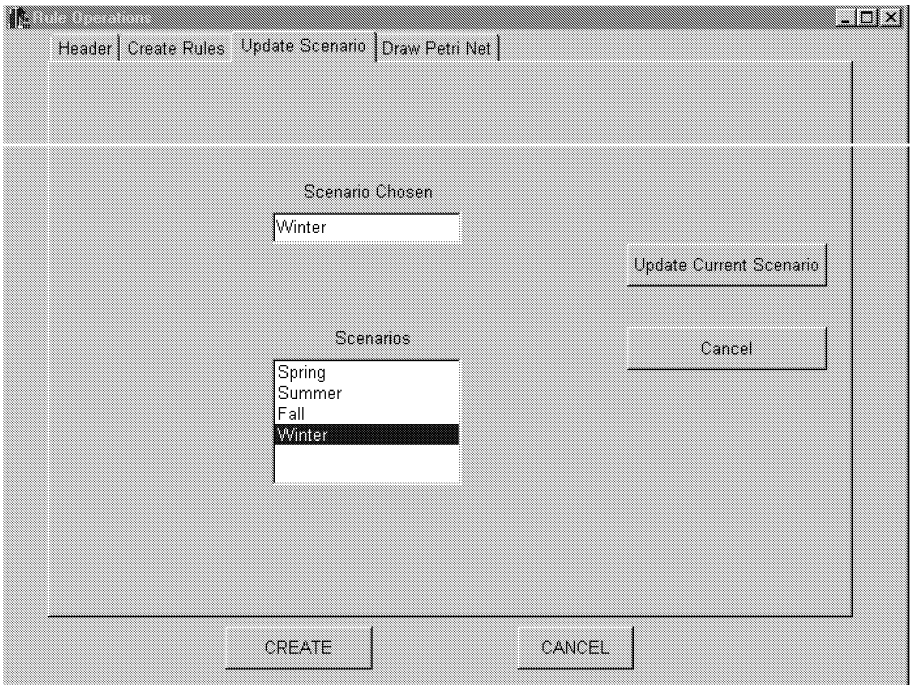
THEN ACTION:
```
forecast.weather_event(snow)
```

RULE TYPE: (● SCENARIO) ( ) ACTIVE ( ) DEDUCTIVE

Scenario Name:

Attributes	Object Attributes	Fuzzy Terms
cloud	wind_value	north
humidity	wind_dr	northwest
pressure	prev_wind_va	west
temperature	prev_wind_dr	southwest
wind	wind_dr_ch_v	south

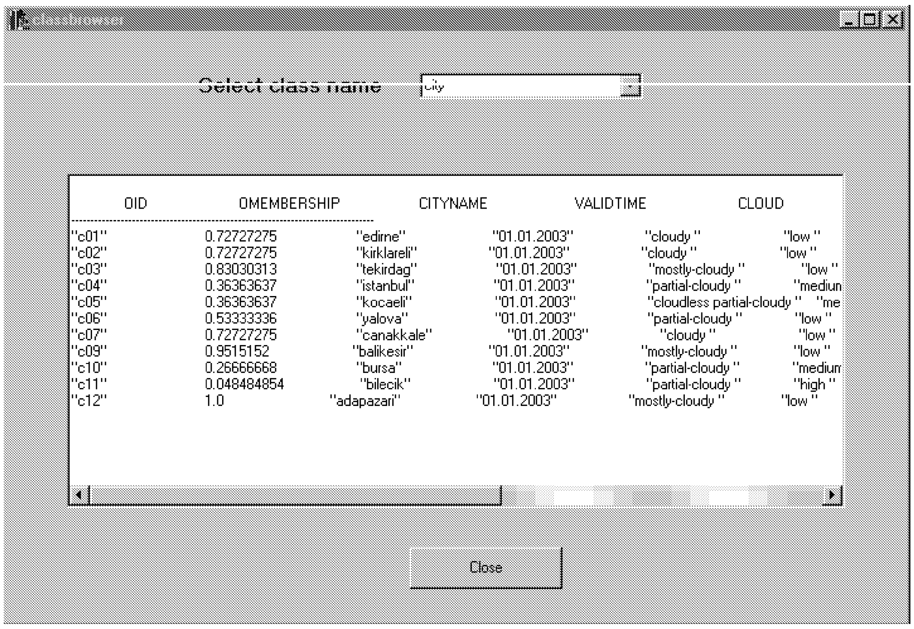
Figure 6.15: Scenario Rule Definition - Rule operations (4)



Scenario Chosen: Winter

Scenarios:
Spring
Summer
Fall
Winter

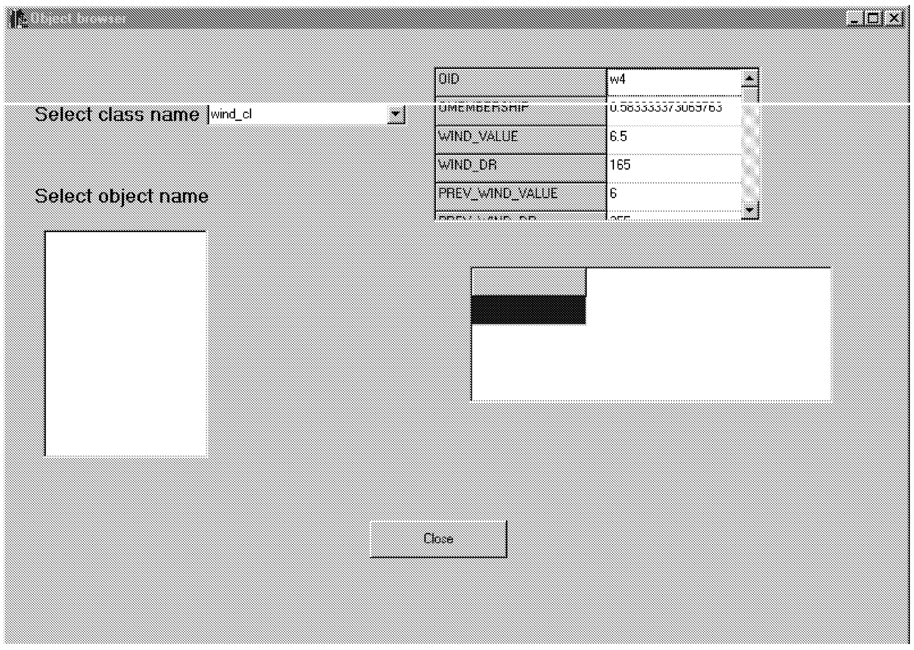
Figure 6.16: Updating Current Scenario - Rule operations (5)



Select class name: City

```
     OID              OMEMBERSHIP           CITYNAME              VALIDTIME              CLOUD
--------------------------------------------------------
"c01"         0.72727275        "edirne"         "01.01.2003"        "cloudy "               "low "
"c02"         0.72727275        "kirklareli"     "01.01.2003"        "cloudy "               "low "
"c03"         0.83030313        "tekirdag"        "01.01.2003"       "mostly-cloudy "          "low "
"c04"         0.36363637        "istanbul"       "01.01.2003"        "partial-cloudy "        "mediun
"c05"         0.36363637        "kocaeli"         "01.01.2003"       "cloudless partial-cloudy "  "me
"c06"         0.53333336        "yalova"          "01.01.2003"       "partial-cloudy "        "low "
"c07"         0.72727275        "canakkale"         "01.01.2003"       "cloudy "               "low "
"c09"         0.9515152        "balikesir"       "01.01.2003"        "mostly-cloudy "         "low "
"c10"         0.26666668        "bursa"          "01.01.2003"        "partial-cloudy "        "mediun
"c11"         0.048484854        "bilecik"        "01.01.2003"       "partial-cloudy "        "high "
"c12"         1.0           "adapazari"      "01.01.2003"       "mostly-cloudy "         "low "
```

Figure 6.17: Class browser



Select class name wind_cl

Select object name

OID	w4
UMEMBERSHIP	0.583333373069763
WIND_VALUE	6.5
WIND_DR	165
PREV_WIND_VALUE	6

Figure 6.18: Object browser

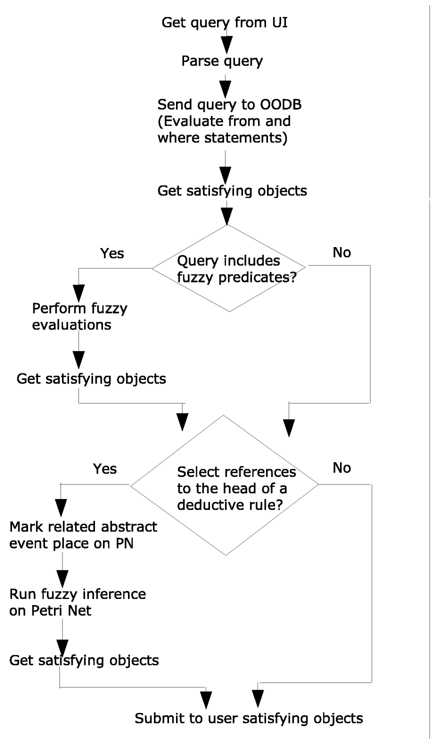
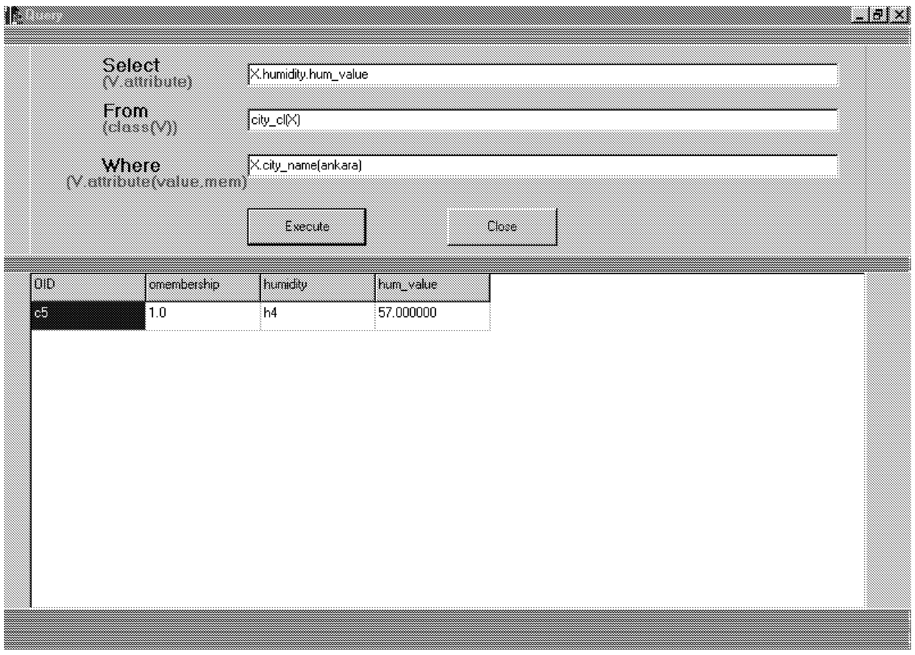

Get query from UI
Parse query
Send query to OODB (Evaluate from and where statements)
Get satisfying objects
Yes
Query includes fuzzy predicates?
No
Perform fuzzy evaluations
Get satisfying objects
Yes
Select references to the head of a deductive rule?
No
Mark related abstract event place on PN
Run fuzzy inference on Petri Net
Get satisfying objects
Submit to user satisfying objects

Figure 6.19: Steps in Evaluation of a User Query



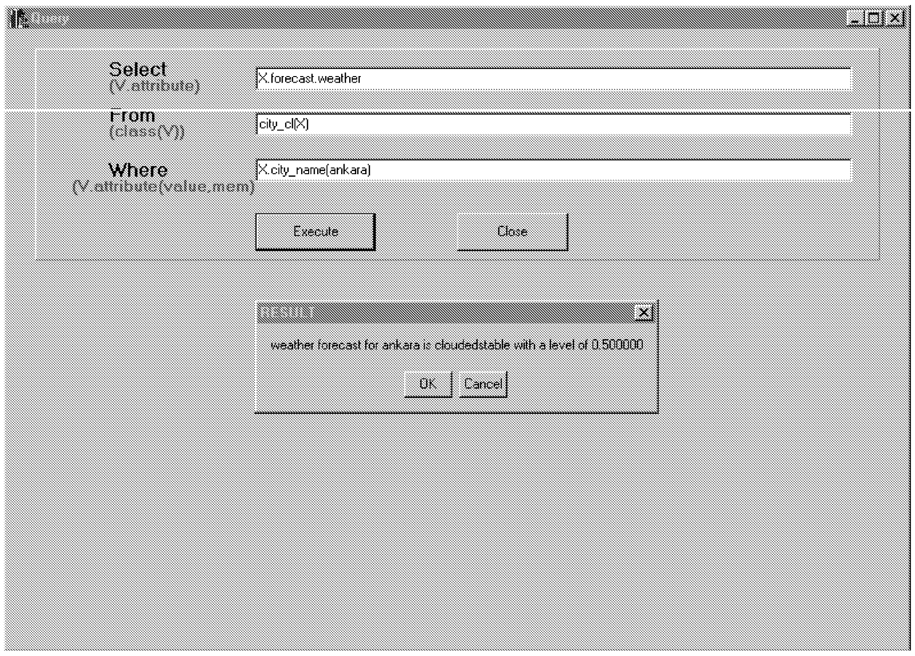
Select (V.attribute): X.humidity.hum_value

From (class(V)): city_cl(X)

Where (V.attribute(value,mem)): X.city_name(ankara)

OID	omembership	humidity	hum_value
c5	1.0	h4	57.000000

Figure 6.20: Simple Query



Select (V.attribute): X.forecast.weather

From (class(V)): city_cl(X)

Where (V.attribute(value,mem)): X.city_name(ankara)

weather forecast for ankara is cloudedstable with a level of 0.500000

Figure 6.21: A Query Requiring Fuzzy Inference on FPN

query and just needs to access the attributes of the related object (Figure 6.20. However, if the *weather* attribute of the *forecast* attribute of a *city* is queried, this requires evaluation of deductive rules using fuzzy Petri Net based inference mechanism (Figure 6.21). Figure 6.19 shows the steps in evaluation of a user query.

6.3.6 Computational Complexities of the Algorithms

6.3.6.1 Construction of the FPN

In Algorithm given at Figure 6.22, first, n number of rules are processed. In each of them, event, condition and action parts get examined (Figures 6.23, 6.24 and 6.25). Suppose that average number of patterns in the event part is p_e, and in the condition part is p_c. We all have 1(one) action. It is negligible. Possible number of operations in this part is: n x (p_e + p_c).

Then, we construct fuzzy inference groups (Figure 6.26). First, we determine the rules which are in the same fuzzy inference group. For that, for each rule, we go over the other rules, which requires n^2 operations. Then we create primitive events for each rule. Let's say we have p_e average number of patterns in the event part of a rule. This requires n x p_e number of operations. Then for each distinct fuzzy inference group we create a combined action place, a fuzzy action place, and for each rule in the group set a link to these places. Let's say we have g number of distinct inference group, and in the average we have n_g number of rules in each inference group. This requires takes 2 x g x n_g number of operations. Possible number of operations in this part is: n^2 + n x p_e + 2 x g x n_g

Finally, we construct transitions (Figure 6.27). For each primitive event, we check the fuzzy primitive events of the rule it belongs. If we have e_p number of primitive events, and we have n number of rules they belong, each one of them having p_e average number of patterns in their event part, this operation requires e_p x n x p_e operations. Then, for each fuzzy composite condition, we initiaize related links. If we have c_e number of composite events, each having p_e number of component places at the worst case, this requires c_e x p_e operations.

```
Input: rules entered by the user
Output: a set of FPN places and transitions
Begin
while (there are still some rules) do
    // attrlist contains the attributes event_text, condition_text, action_text,
    // threshold, fuzzyinferencegroup and the links on the FPN
    create_rule(rule_type, class_name, rule_id, attrlist)

    examine_event(rule_id, rule_classname, event_text)
    examine_condition(rule_id, rule_classname, condition_text)
    examine_action(rule_id, rule_classname, action_text)
end while
construct_fuzzy_inference_groups()
construct_transitions()
End
```

Figure 6.22: Construct_FPN Algorithm

```
Input: rule_id $r_i$, rule_classname, event_text
Output: a fuzzycompositeevent place and its component fuzzyprimitievent places
Begin
tokenize(event_text,tokecount,tokenarray)
if ($tokencount > 1$) then
  initialize tempcount to 0
  create a new place object of type fuzzycompositeevent
  put an entry in the place list for this place
  update the attributes of the related FPN place
  update the attributes of the related rule
  while ($tempcount \leq tokencount$) do
    if ($NULL == unifies(tokenarray|tempcount|)$) then
      create a new place object of type fuzzyprimitiveevent
      put an entry in the place list for this place
    end if
    update the attributes of the related FPN place
    update the attributes of the related rule
    increment tempcount
  end while
else
  if ($tokencount == 1$) then
    if ($NULL == unifies(tokenarray|tempcount|)$) then
      create a new place object of type fuzzyprimitiveevent
      put an entry in the place list for this place
    end if
    update the attributes of the related FPN place
    update the attributes of the related rule
  end if
end if
End
```

Figure 6.23: Examine_Event Algorithm

```
Input: rule_id $r_i$, rule_classname, condition_text
Output: a fuzzycompositecondition place and its component
fuzzyprimiticondition places
Begin
tokenize(condition_text,tokencount,tokenarray)
if ($tokencount > 1$) then
    initialize tempcount to 0
    create a new place object of type fuzzycompositecondition
    put an entry in the place list for this place
    update the attributes of the related FPN place
    update the attributes of the related rule
    while ($tempcount \leq tokencount$) do
        create a new place object of type fuzzyprimitivecondition
        put an entry in the place list for this place
        update the attributes of the related FPN place
        update the attributes of the related rule
        increment tempcount by 1
    end while
else
    if ($tokencount == 1$) then
        create a new place object of type fuzzyprimitivecondition
        put an entry in the place list for this place
        update the attributes of the related FPN place
        update the attributes of the related rule
    end if
end if
End
```

Figure 6.24: Examine_Condition Algorithm

```
Input: rule_id r_i, rule_classname, action_text
Output: a fuzzyclippedaction place
Begin
create a new place object of type fuzzyclippedaction
put an entry in the place list for this place
update the attributes of the related FPN place
update the attributes of the related rule
End
```

Figure 6.25: Examine_Action Algorithm

As for the composite conditions, applying the same approach, we require c_c x p_c operations. For each rule, we create trigger condition transitions to the fuzzy primitive conditions of the rule. This takes n x p_c number of operations in the worst case. For each rule in an inference group, we set the related links from their action part to the combined action. This takes g x n_g operations. In order to create event generating transitions, we check each fuzzy primitive condition with the primive events and we check each fuzzy action with the primive events. In worst case, this takes e_p x (p_c+ n) operations. Therefore, a possible number of operations in this part is :

e_p x n x p_e + c_e x p_e + c_c x p_c + n x p_c + g x n_g + e_p x (p_c+ n)

6.3.6.2 Determining true cycles

In Algorithm given at Figure 5.4 in Chapter 5, in worst case, we have to process the while loop n times (n is the number of rules). In each cycle, we have to obtain the matrix multiplications of two matrices each of which is n by n. Matrix multiplication requires n^3 operations. So totally determining true cycles requires n^4 operations, O(n^4).

6.3.6.3 Checking the Confluence

In Algorithm given at Figure 5.5 in Chapter 5, in worst case, each rule constitutes a different inference group, so resulting with n number of inference groups (n is the number of rules). Checking inference groups with each other (2 for loops) requires n^2 operation, O(n^2).

6.3.6.4 Inference Algorithm

In Algorithm given at Figure 5.6 in Chapter 5, in worst case, we have 1(one) rule in each inference group, and each rule gets triggered. So, the outermost while loop should be performed n times. In each cycle we go over the transitions (let we have t number of transitions), and for each transition, we go over the places 2(two) times, one for the input places, one for the output places. So a total of 2 x n x t x p operations gets performed. Which is O(ntp).

```
Begin
for (each rule $r_i$) do
  for (each rule $r_j$) do
    if $r_i$ unifies with $r_j$ then they're in the same inference group
  end for
  if (fuzzy inference group of $r_i$ not processed) then
    for (each different fuzzy domain in the event of $r_i$) do
      create a new place object of type primitive event
      put an entry in the place list for this place
      update the attributes of the related FPN place
    end for
    create a new place object of type combined action
    put an entry in the place list for this place
    update the attributes of the related FPN place
    for (each rule $r_k$ in the fuzzy inference group of $r_i$) do
      set a link to the created place
    end for
    create a new place object of type fuzzy action
    put an entry in the place list for this place
    update the attributes of the related FPN place
    for (each rule $r_k$ in the fuzzy inference group of $r_i$) do
      set a link to the created place
    end for
  end if
  create a new place object of type strengthened event
  put an entry in the place list for this place
  update the attributes of the related FPN place
end for
End
```

Figure 6.26: Construct_Fuzzy_Inference_Groups Algorithm

```
Begin
for (each primitive event $p_i$) do
  for (each fuzzy primitive event $p_j$) do
    if they are related then create fuzzify event, update links
  end for
end for
for (each fuzzy composite event $p_i$) do
  create composition transition, update the links
end for
for (each fuzzy composite condition $p_i$) do
  create composition transition, update the links
end for
for each rule $r_i$ do create strength transition, update the links
for (each fuzzy primitive condition $p_j$ of $r_i$) do
  create trigger condition transition, update the links
end for
create clipping transition, update the links
if (fuzzy inference group of $r_i$ is not processed) then
  create combine concurrent actions transition, update the links
  for (each rule $r_j$ in the $r_i$'s fuzzy inference group do
    create obtain fuzzy action transition, update the links
  end for
end if
for (each place pi either a fuzzy action or fuzzy primitive condition) do
  for (each primitive event $p_j$) do
    if they are related then create event generating transition
  end for
end for
End
```

Figure 6.27: Construct_Transitions Algorithm

CHAPTER 7

AN APPLICATION EXAMPLE: WEATHER FORECASTING

The Weather forecasting is chosen as a case study for implementation of the model developed in this thesis. Therefore, weather forecasting is intoduced shortly in this chapter.

7.1 What is Included in Our Forecast

We have interviewed the weather forecast experts from the "*Turkish State Meteorological Service*" to understand the domain of weather forecasting application and draw its requirements [76]. What they say is that it is very hard and complicated to device a forecast system covering all parts of Turkey due to Turkey's diverse geographical features. Therefore, we have concentrated on some part of Turkey and studied on the "weather lore" for the Central Anatolia.

The atmospheric elements we consider are:

- Pressure,
- Temperature,
- Relative Humidity (we'll simply call it humidity),
- Wind, and

- Cloudiness.

Our rules consider the change on the above parameters, such as value changes, direction changes or velocity changes, etc. There exists two level forecast in our model. In the first level we forecast weather, which could be one of the followings:

- clear,
- clear few,
- clouded instable, or
- clouded stable.

In the second level, we forecast the weather event according to the output of the first level forecast together with the newly changing parameters on the atmospheric elements. In this level, forecasted weather event can be any of the followings:

- rain,
- shower,
- snow,
- hail, or
- fog.

Temperature change is also forecasted in the second level. It is again determined from the first level forecast results together with the newly changing parameters on the atmospheric elements. The forecasted temperature change can be any of the followings:

- increase on temperature,
- decrease on temperature, or
- no change on temperature.

7.1.1 Season Scenarios

In order to put the effect of seasons, we partition our rules of weather according to seasons. That is, each season functions like a scenario (Section 3.1) for us. By this way, we give more emphasis on some rules according to the season. Table 7.1 partitions our rules of weather, weather event and temperature change, according to seasons.

Table 7.1: Season Scenarios

Scenario	Weather	Weather Event	Temperature Change
Summer	Clear, Clear few	-	Increase
Winter	Clouded stable	Rain, Fog, Snow	Decrease
Spring, Fall	Clouded instable	Rain, Hail, Shower	Increase

7.2 What "Weather Lore" is

The experts who forecast weather based on the direct observation are the real experts of the weather forecasting. These experts are the people who live permanently in the country or whose livehood depend on their ability to predict changing local weather like anglers, hunters, farmers, sailors, mountaineers, campers, ballonists. Many of their insights have been passed from one generation to another as *weather lore* and collected in various weather almanacs.

An advantage of weather lore is that it is expressed in terms of set of rules of thumb that embody fuzzy input-output relationships [39]. [48] gives the following example:

When the barometer falls fast, temperature is less than -1°C, wind is blowing from south and east, snow is expected within 12 to 24 hours.

This type of commonsense rules related to the atmospheric conditions may be valid only in one locale, or during one season of the year, or only at sea.

Two sorts of applications of fuzzy logic in weather prediction are expert systems and case based reasoning systems. As far as the fuzzy expert systems for weather prediction are considered, Maner et al. [53] build a weather prediction system called WXSYS, which uses the simple weather prediction rules from experts and weather almanacs, and implement these using fuzzy logic rule base. Systems build by Sujthjorn et al. [70] and Murtha [61] predict fog at an airport. Hansen [41] build a system for critiquing marine forecasts, which is called SIGMAR. Instead of processing a series of fuzzy rules, it measures the similarity between a current marine forecast and the actual marine observations to amend marine forecasts. Hadjimicheal et al. [40] build a fuzzy system called MEDEX, for forecasting gale force winds in the mediterranean. Shao et al [69] build an automated wire-ice prediction model to provide short-period forecasts on the wires. As for the fuzzy case based reasoning systems for weather prediction are concerned, Tag et al. [72] and Bardossy et al. [8] use fuzzy logic to automate the recognition of patterns of upper air wind flow. Hansen et al. [42] study ceiling and visibility at airport. In that, given a present incomplete weather case to predict for, they use a fuzzy k-nn algorithm to find similar past weather cases in a huge weather archive to make predictions from.

Our study resides on the side of fuzzy expert systems for weather prediction. However, the above studies on the same category differ from us on a couple of aspects. First, type of the rules used by other systems are not active. Also they do not put the effect of seasonal changes in their inference. In addition some forecasts may be occasionally self contradicting. i.e., forecasting both wet and dry weather at the same time without giving an order [53].

7.3 Why Fuzzy Logic is Suitable for Predicting the Weather

Language used in conventional forecasts are inherently and intentionally fuzzy. Consider, these commonly occuring phrases: "mostly clear", "toward midnight", "becoming cooler overnight", "winds becoming westerly", "light to moderate snow", "increasing high cloudiness", etc. All of these values can be naturally represented by fuzzy sets. The weather domain meets the general conditions under which fuzzy solution is thought to be appropriate. It is a domain where:

- approximate solutions are acceptable.
- the value and range of important variables, such as wind direction, amount of cloud cover, temperature, are represented numerically.
- input-output relationships exist but may not be well-defined or even not consistent. Depending on the season and local geography, the same input weather conditions can lead to different output weather forecasts.
- either no mathematical formula is available that can produce the desired result or there is a formula but it is too complex to produce results in real time. It is probably fair to say that, the best algorithms, when applied to reasonable computational models of the atmosphere, probably exceed the capacity of all but the largest and fastest computers [53].
- there is a wish to make more use of available information, but doing so would make an crisp algoritmic solution too complex.

7.4 Active Rules and FPN Construction

7.4.1 1st Level Rules - Rules for Forecasting the Weather

For that we use the following rules:

```
R1: (CLEAR WEATHER)
  ON    pressure.pres_ch_dir(increasing),
        wind.wind_dr(west northwest),
        humidity.hum_ch_dir(decreasing),
        cloud.cover(brokensky),
        cloud.base_ch_dir(increasing)
  IF    pressure.pres_ch_vel(fast),
        wind.prev_wind_dr(south),
        humidity.hum_ch_vel(fast),
        cloud.prev_cover(overcast),
        cloud.base_ch_vel(fast)
  THEN  forecast.weather(clear)

R2: (CLEAR_FEW WEATHER)
  ON    pressure.pres_ch_dir(increasing),
        cloud.cover(brokensky few),
        cloud.base_ch_dir(increasing),
        humidity.hum_ch_dir(decreasing),
        wind.wind_dr(north northwest)
  IF    pressure.pres_ch_vel(slow),
        wind.wind_value(breeze),
        wind.prev_wind_dr(south southwest),
        cloud.prev_cover(overcast cloudy)
  THEN  forecast.weather(clearfew)

R3: (CLOUDED_INSTABLE WEATHER)
  ON    pressure.pres_ch_dir(decreasing),
        humidity.hum_ch_dir(increasing),
        wind.wind_value(mediumstrong strong),
        cloud.orient(vertical),
        cloud.base_ch_dir(decreasing),
        temperature.temp_ch_dir(increasing)
  IF    pressure.pres_ch_vel(fast),
        humidity.hum_ch_vel(fast),
        wind.prev_wind_value(breeze mediumstrong)
  THEN  forecast.weather(cloudedinstable)

R4: (CLOUDED_STABLE WEATHER)
  ON    pressure.pres_ch_dir(decreasing),
          humidity.hum_ch_dir(increasing),
          temperature.temp_ch_dir(increasing),
          cloud.base_ch_dir(decreasing),
          cloud.orient(horizontal)
     IF   pressure.pres_ch_vel(slow),
          humidity.hum_ch_vel(slow),
          temperature.temp_ch_vel(slow),
          cloud.base_ch_vel(slow),
          wind.wind_dr(southeast south)
     THEN forecast.weather(cloudedstable)
```

The properties of **forecasting the weather** is listed below:

- These rules track the changes on the following atmospheric elements: *wind value, wind direction, cloud orientation, cloud base, cloud cover, humidity value, temperature value* and *pressure value.* We consider attribute changes made by user as if they come from sensor.

- Individual changes on the above elements has no meaning. Only if a couple of changes happening together can allow us to determine the forthcoming weather. For example, only if pressure is increasing, cloud cover is changing to broken sky or few, cloud base is increasing, humidity is decreasing and wind direction is changing to north or northwest, we can say that these are an indication of a possible *clear few weather.* Therefore, we wait for all the changes to occur, i.e. we collect the changes on the attributes that the rules R_1 to R_4 use.

- If all changes occur, rules R_1 to R_4 are triggered at the same time. They should be executed concurrently using a fuzzy inference mechanism.

- In that fuzzy inference, the effect of the season (i.e. current scenario) should also be considered. Season determines the state of the application environment, gives more importance to some rules and prunes some others. This is found with similarities of rules to the current season or scenario. In *Winter*, for example, we expect a *clouded stable* weather, however, in *Summer* we expect a *clear* or *clear few* weather.

More concretely, we perform the following steps during the fuzzy inference:

Step1 Once we have the values of atmospheric elements, which take part in triggering the rules R_1 to R_4, we **fuzzify** them, and obtain **fuzzy primitive events**. For example, if we have the value of wind changing from 6 to 6.5, we need to find to what degree it is becoming *medium* or *medium strong* as R_3 requires.

Step2 We perform the **composition of events**, which are fuzzified at Step1. For that we use min for AND and max for OR operator and obtain the composite event.

Step3 Using the similarity of each rule and the composite events calculated at Step2, we obtain the **event strenghts** for each rule.

Step4 We calculate the condition matching degrees for each **primitive condition**. For that we fuzzify the attribute values which are checked in condition parts. For example, R_3 checks whether the previous wind value was *breeze* or *medium strong*.

Step5 We perform the **composition of conditions**, which are fuzzified at Step4. For that we use min for AND and max for OR operator and obtain the **composite condition**.

Step6 Using the event strenghts calculated at Step3 and composite conditions calculated at Step5, we calculate the **clipping value** for each rule.

Step7 We **combine the clipping values** of all rules calculated at Step6.

Step8 We find the **overall fuzzy action** within the combined actions in Step7, who has the maximum clipping value within others. For example, we forecast a *clear weather* with a level of 0.5.

These FPN execution items are all shown on top of the FPN figures drawn.

7.4.2 2nd Level Rules

7.4.2.1 Rules for Forecasting the Weather Event

```
R5: (RAIN)
  ON   forecast.weather(cloudedstable),
       wind.wind_dr(south southwest),
       wind.wind_value(mediumstrong),
       cloud.color(grey)
  IF   temperature.temp_value(chill warm hot),
       wind.prev_wind_dr(north),
       wind.wind_dr_ch_vel(slow),
       wind.prev_wind_value(breeze),
       wind.wind_ch_vel(slow),
       cloud.prev_color(white)
  THEN forecast.weather_event(rain)

R6: (SHOWER)
  ON   forecast.weather(cloudedinstable),
       cloud.color(darkgrey)
  IF   wind.prev_wind_value(calm),
       cloud.prev_color(grey)
  THEN forecast.weather_event(shower)

R7: (SNOW)
  ON   forecast.weather(cloudedstable),
       wind.wind_dr(north),
       cloud.color(grey)
  IF   temperature.temp_value(cold),
       wind.prev_wind_dr(south),
       cloud.prev_color(white)
  THEN forecast.weather_event(snow)
```

```
R8: (HAIL)
  ON   forecast.weather(cloudedinstable),
       cloud.color(dark)
  IF   temperature.temp_value(hot),
       wind.prev_wind_value(breeze),
       cloud.prev_color(grey)
  THEN forecast.weather_event(hail)

R9: (FOG)
  ON   forecast.weather(cloudedstable)
  IF   wind.wind_value(calm breeze)
  THEN forecast.weather_event(fog)
```

The properties of **forecasting the weather event** is listed below:

- These rules track the changes on the following atmospheric elements: *wind value, wind direction, cloud color and pressure value* together with a weather being forecasted with rules R_1 to R_4. Therefore, without rules R_1 to R_4 forecasting a weather, these atmospheric changes has no meaning. Or we can say that there is an early indication of a weather event to forecast using rules R_5 to R_9 with a weather forecasted using rules R_1 to R_4.

- Individual changes on the above elements has no meaning. Only if a couple of changes happening together can allow us to determine the forthcoming weather event. Again, we wait for all the changes to occur, i.e. we collect the changes on the attributes that the rules R_5 to R_9 use.

- If all changes occur, rules R_5 to R_9 are triggered at the same time. They should be executed concurrently using a fuzzy inference mechanism.

- In that fuzzy inference, the effect of the season (i.e. current scenario) is again considered. In *Winter*, for example, we expect a *rain*, *fog* or *snow* as a weather event, however, in *Spring* or *Fall* we expect *Rain*, *Hail* or *Shower* as a weather event.

The steps performed in fuzzy inference are same as the ones performed during forecasting a weather. However, what is forecasted now is a weather event.

7.4.2.2 Rules for Forecasting the Temperature Change

```
R10: (TEMPERATURE INCREASE)
  ON   forecast.weather(clear clearfew cloudedinstable),
       pressure.pres_ch_dir(decreasing),
       wind.wind_dr(south southwest)
  IF   wind.prev_wind_dr(north northwest northeast)
  THEN forecast.temperature_change(increase)

R11: (TEMPERATURE DECREASE)
  ON   forecast.weather(cloudedstable),
       pressure.pres_ch_dir(increasing),
       wind.wind_dr(north northwest northeast)
  IF   wind.prev_wind_dr(south southwest)
  THEN forecast.temperature_change(decrease)

R12: (NO TEMPERATURE CHANGE)
  ON   forecast.weather(clear clearfew cloudedinstable cloudedstable)
  IF   pressure.pres_ch_dir(nochange),
       wind.wind_dr(west),
       wind.wind_value(breeze)
  THEN forecast.temperature_change(nochange)
```

Figures 7.1, 7.2, 7.3 show the parts of the FPN constructed for these active rules of weather forecasting.

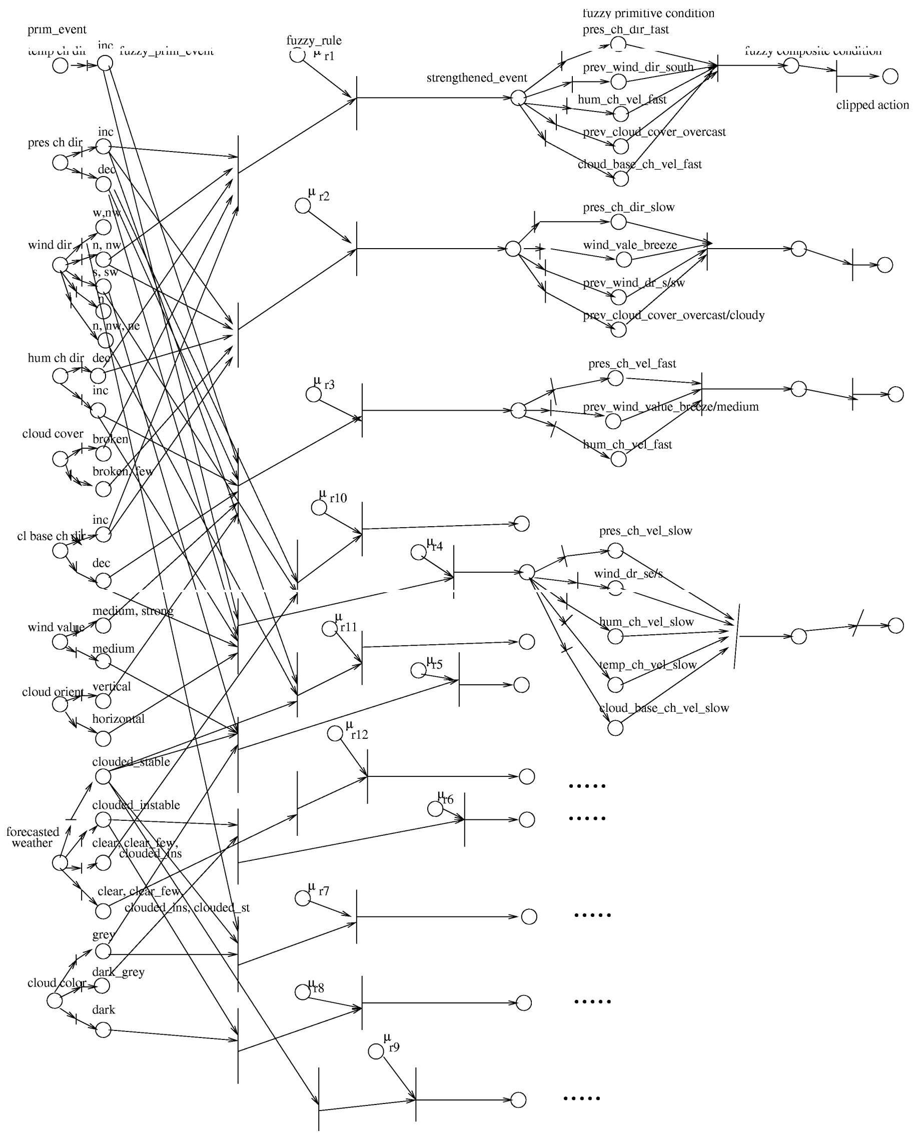


Figure 7.1: FPN constructed for the weather forecasting

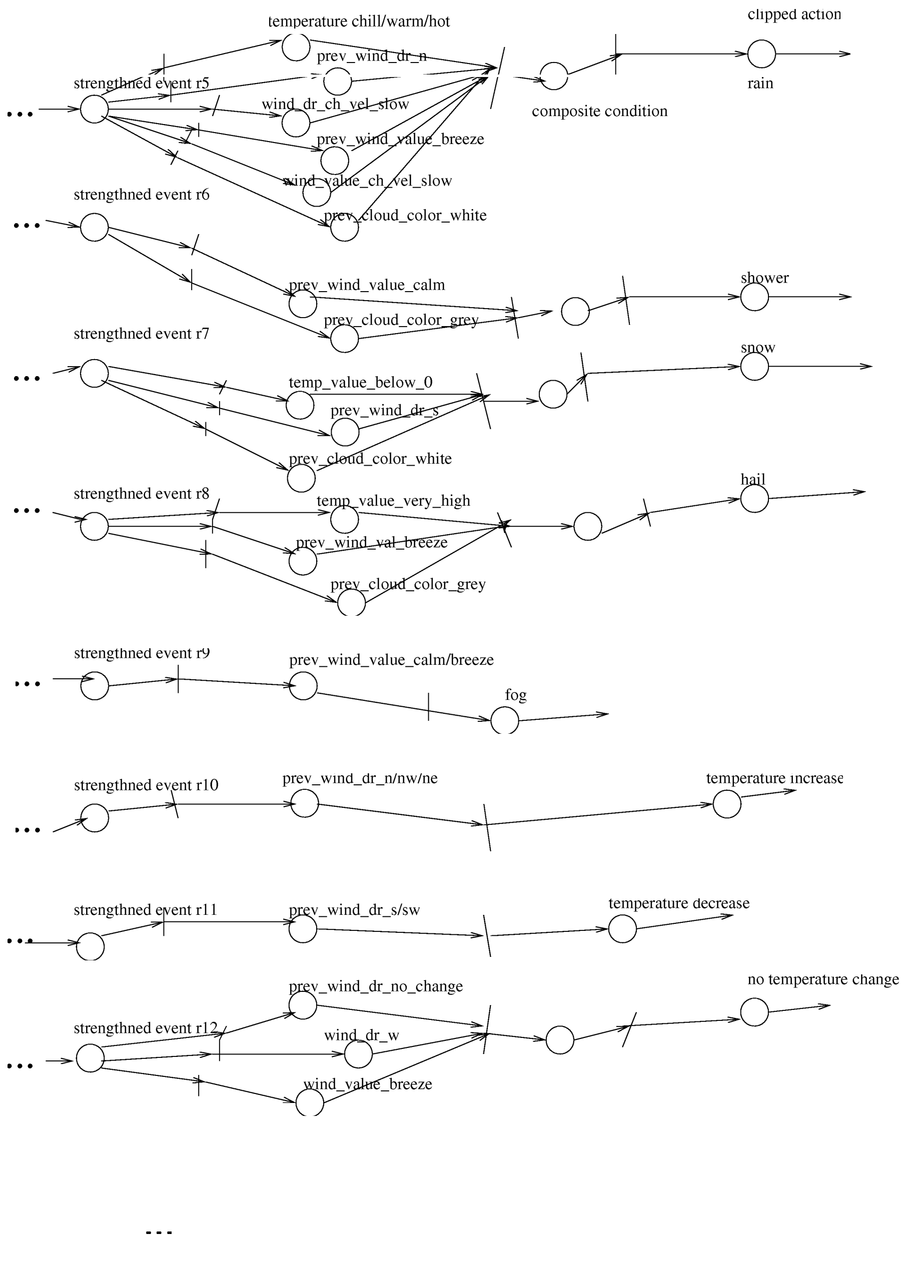


Figure 7.2: FPN constructed for the weather forecasting (cont)

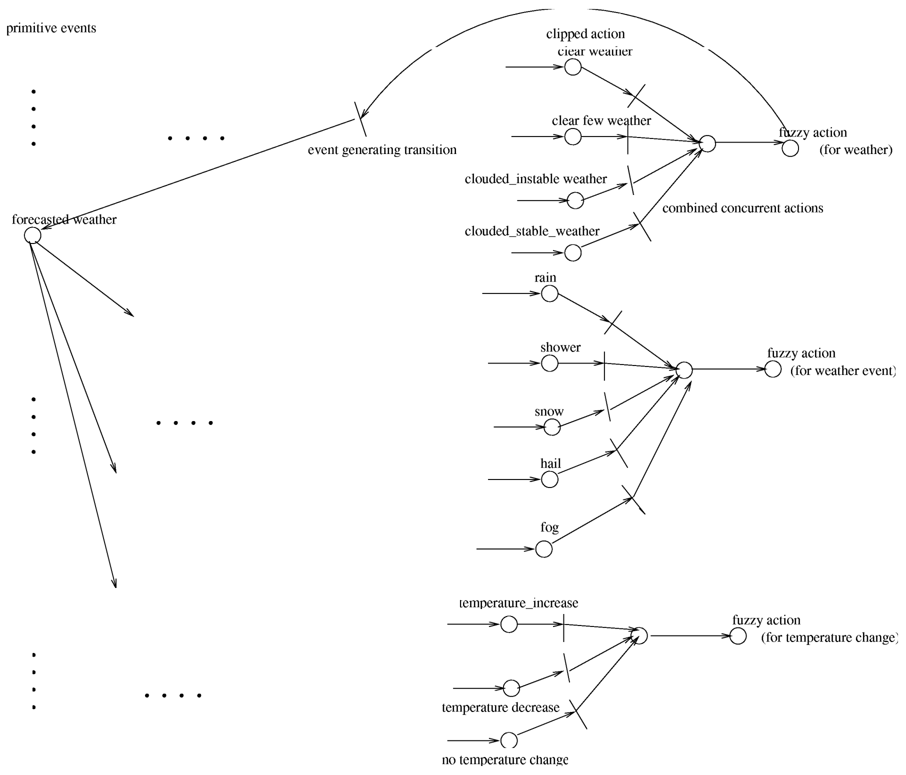


Figure 7.3: FPN constructed for the weather forecasting (cont)

7.5 Deductive Rules and FPN Construction

Deductive rules are the ones having no event part. However, internally all rules are of active kind. Therefore, system converts the deductive rules to active rules having an abstract kind of event.

7.5.1 Rules for querying the weather forecast

```
R01: (CLEAR WEATHER)
  IF   pressure.pres_ch_dir(increasing),
       wind.wind_dr(west northwest),
       humidity.hum_ch_dir(decreasing),
       cloud.cover(brokensky),
       cloud.base_ch_dir(increasing)
       pressure.pres_ch_vel(fast),
       wind.prev_wind_dr(south),
      humidity.hum_ch_vel(fast),
      cloud.prev_cover(overcast),
      cloud.base_ch_vel(fast)
  THEN forecast.weather(clear)

R02: (CLEAR_FEW WEATHER)
  IF   pressure.pres_ch_dir(increasing),
       cloud.cover(brokensky few),
       cloud.base_ch_dir(increasing),
       humidity.hum_ch_dir(decreasing),
       wind.wind_dr(north northwest)
       pressure.pres_ch_vel(slow),
       wind.wind_value(breeze),
       wind.prev_wind_dr(south southwest),
       cloud.prev_cover(overcast cloudy)
  THEN forecast.weather(clearfew)

R32: (CLOUDED_INSTABLE WEATHER)
  IF   pressure.pres_ch_dir(decreasing),
       humidity.hum_ch_dir(increasing),
       wind.wind_value(mediumstrong strong),
       cloud.orient(vertical),
       cloud.base_ch_dir(decreasing),
       temperature.temp_ch_dir(increasing)
       pressure.pres_ch_vel(fast),
       humidity.hum_ch_vel(fast),
       wind.prev_wind_value(breeze mediumstrong)
  THEN forecast.weather(cloudedinstable)

R42: (CLOUDED_STABLE WEATHER)
  IF   pressure.pres_ch_dir(decreasing),
        humidity.hum_ch_dir(increasing),
        temperature.temp_ch_dir(increasing),
        cloud.base_ch_dir(decreasing),
        cloud.orient(horizontal)
        pressure.pres_ch_vel(slow),
        humidity.hum_ch_vel(slow),
        temperature.temp_ch_vel(slow),
        cloud.base_ch_vel(slow),
        wind.wind_dr(southeast south)
  THEN  forecast.weather(cloudedstable)
```

For example, deductive rule R_{01} is converted to the following active rule, which has an abstract kind of event

```
R01: (CLEAR WEATHER)
  ON    raise(forecast.weather)
  IF    pressure.pres_ch_dir(increasing),
        wind.wind_dr(west northwest),
        humidity.hum_ch_dir(decreasing),
        cloud.cover(brokensky),
        cloud.base_ch_dir(increasing)
        pressure.pres_ch_vel(fast),
        wind.prev_wind_dr(south),
        humidity.hum_ch_vel(fast),
        cloud.prev_cover(overcast),
        cloud.base_ch_vel(fast)
  THEN  forecast.weather(clear)
```

7.5.2 Rules for querying the weather event forecast

```
R52: (RAIN)
  IF    forecast.weather(cloudedstable),
        wind.wind_dr(south southwest),
       wind.wind_value(mediumstrong),
       cloud.color(grey)
       temperature.temp_value(chill warm hot),
       wind.prev_wind_dr(north),
       wind.wind_dr_ch_vel(slow),
       wind.prev_wind_value(breeze),
       wind.wind_ch_vel(slow),
       cloud.prev_color(white)
  THEN forecast.weather_event(rain)

R62: (SHOWER)
  IF   forecast.weather(cloudedinstable),
       cloud.color(darkgrey)
       wind.prev_wind_value(calm),
       cloud.prev_color(grey)
  THEN forecast.weather_event(shower)

R72: (SNOW)
  IF   forecast.weather(cloudedstable),
       wind.wind_dr(north),
       cloud.color(grey)
       temperature.temp_value(cold),
       wind.prev_wind_dr(south),
       cloud.prev_color(white)
  THEN forecast.weather_event(snow)

R82: (HAIL)
  IF   forecast.weather(cloudedinstable),
       cloud.color(dark)
       temperature.temp_value(hot),
       wind.prev_wind_value(breeze),
       cloud.prev_color(grey)
  THEN forecast.weather_event(hail)

R92: (FOG)
  IF   forecast.weather(cloudedstable)
       wind.wind_value(calm breeze)
  THEN forecast.weather_event(fog)
```

7.5.3 Rules for querying the temperature change forecast

```
R102: (TEMPERATURE INCREASE)
  IF   forecast.weather(clear clearfew cloudedinstable),
       pressure.pres_ch_dir(decreasing),
       wind.wind_dr(south southwest)
       wind.prev_wind_dr(north northwest northeast)
  THEN forecast.temperature_change(increase)

R112: (TEMPERATURE DECREASE)
  IF   forecast.weather(cloudedstable),
       pressure.pres_ch_dir(increasing),
       wind.wind_dr(north northwest northeast)
       wind.prev_wind_dr(south southwest)
  THEN forecast.temperature_change(decrease)

R122: (NO TEMPERATURE CHANGE)
  IF   forecast.weather(clear clearfew cloudedinstable cloudedstable)
       pressure.pres_ch_dir(nochange),
       wind.wind_dr(west),
       wind.wind_value(breeze)
  THEN forecast.temperature_change(nochange)
```

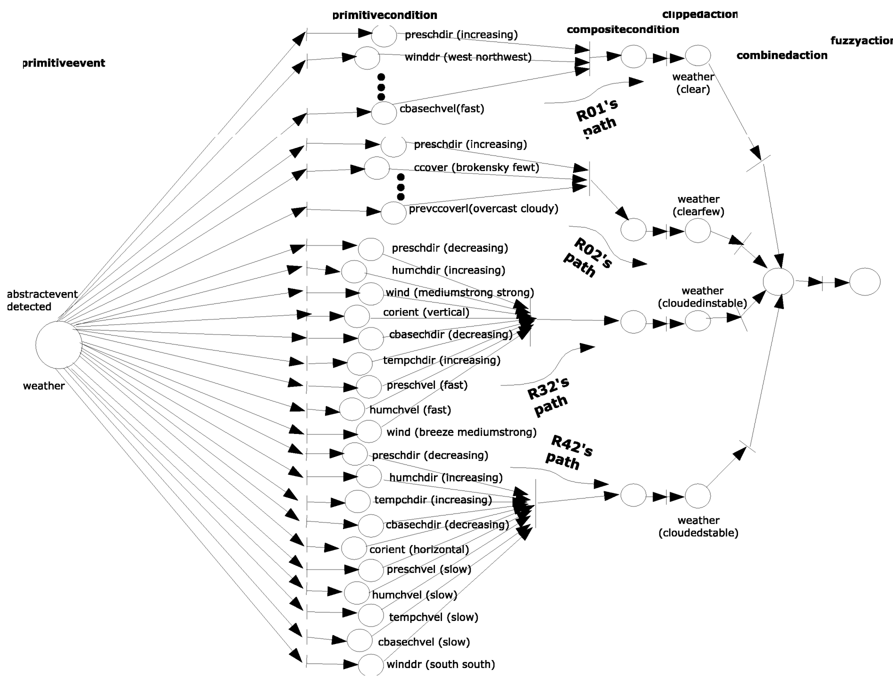


Figure 7.4: FPN constructed for the weather forecasting

Figures 7.4, 7.5, 7.6 show the parts of the FPN constructed for these deductive rules of weather forecasting.

7.6 Examples

In order to keep the system simple, we have defined the rules R_3, R_4, R_5, R_7, R_9, R_{32}, R_{42}, R_{102}, R_{112}, R_{122}. After the definition of these rules, system performs termination and confluence analysis. The result of these analysis for the example rules are shown in Figures 7.7 and 7.8. Petri Net drawn for the example rules in the application program is shown in Figures 7.9 to 7.15. Places corresponding to deductive rules start with a *d* letter while the ones corresponding to active ones start with a *p* letter. Each place is shown with its place type and rules using that place. Table 7.2 lists the abbreviations used for the place types.

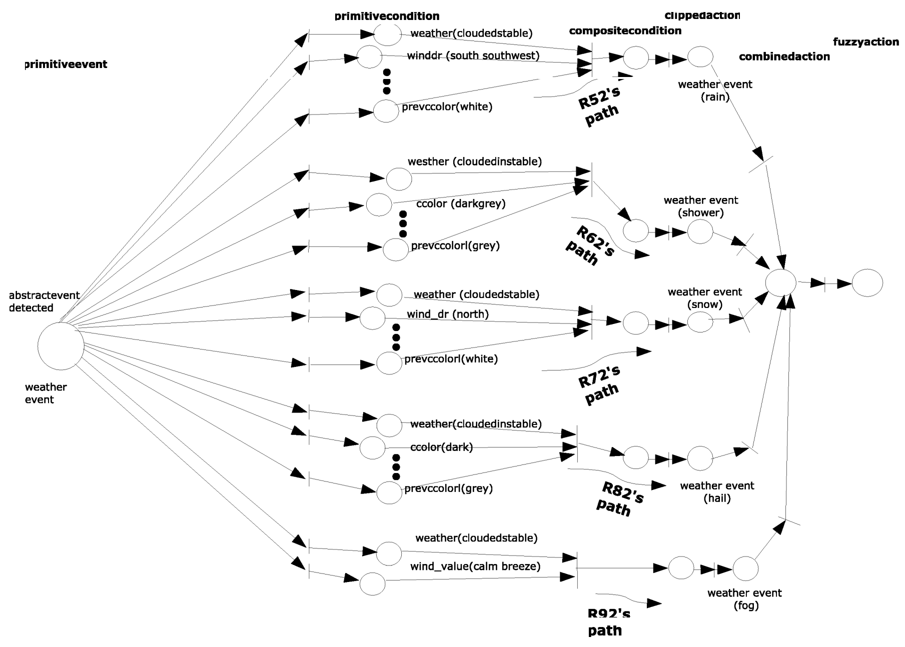


Figure 7.5: FPN constructed for the weather forecasting (cont)

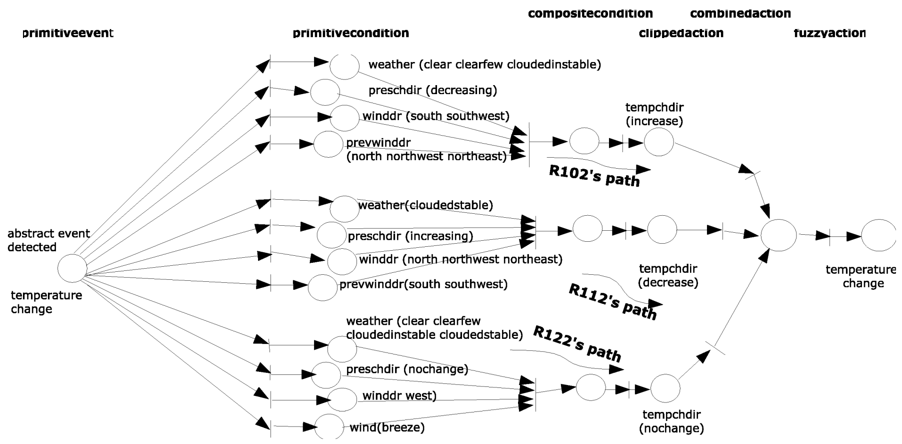
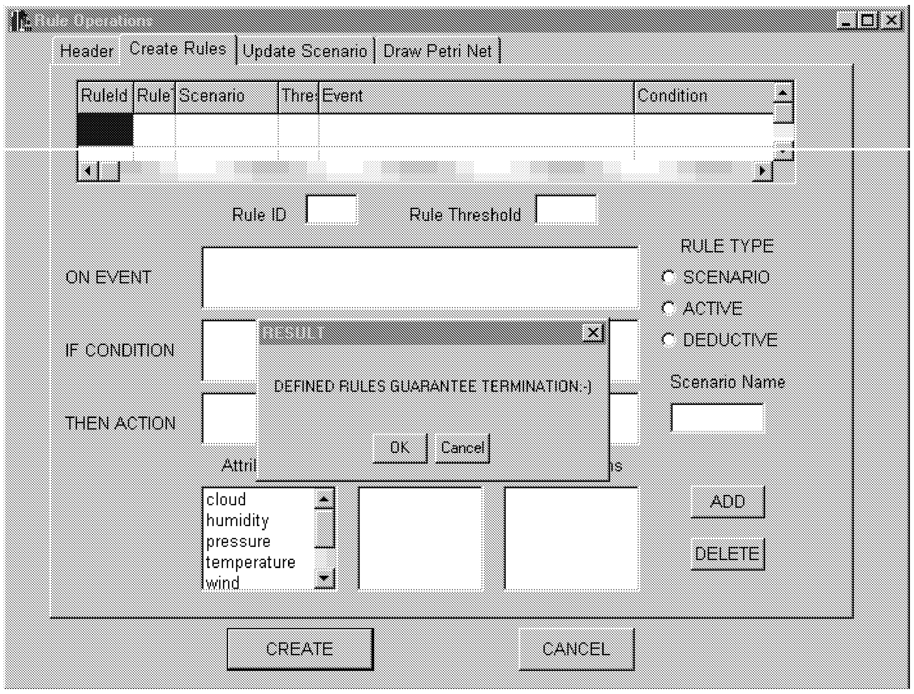


Figure 7.6: FPN constructed for the weather forecasting (cont)

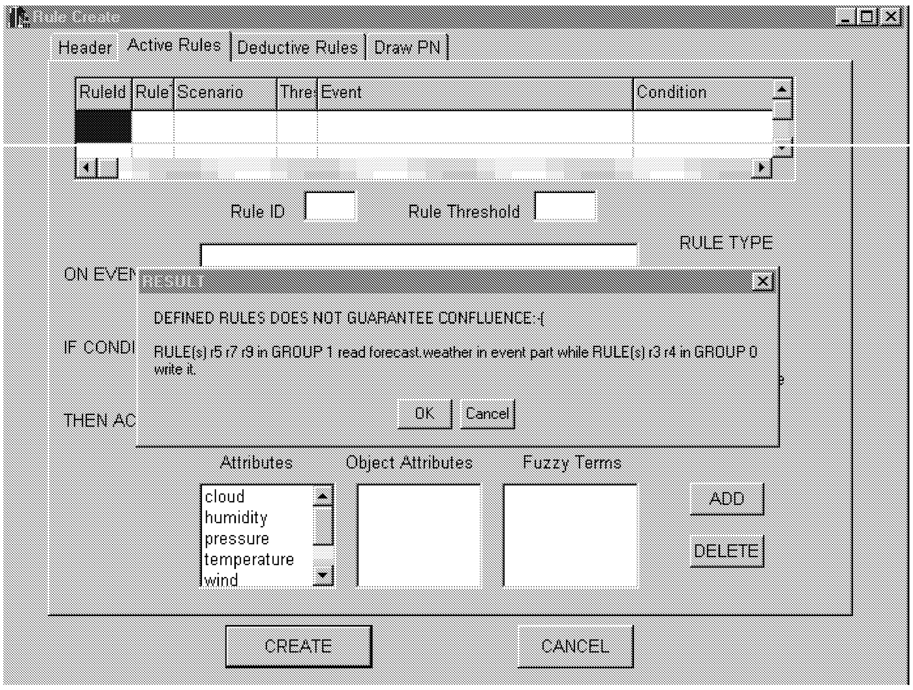
Table 7.2: Abbreviations used in the Petri Net Diagram

Abbreviation	Place Type
pme	primitive event
fpe	fuzzy primitive event
fce	fuzzy composite event
rlp	rule place
ste	strenghtened event
fpc	fuzzy primitive condition
fcc	fuzzy composite condition
cpa	clipped action
cma	combined action
fza	fuzzy action



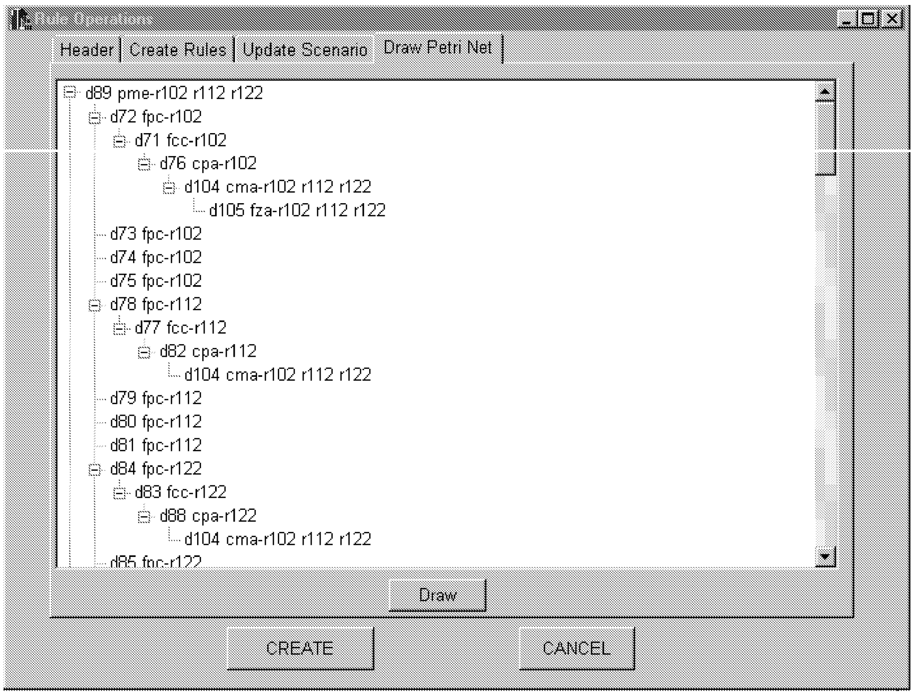
DEFINED RULES GUARANTEE TERMINATION:-)

Figure 7.7: Termination Analysis for the example rules



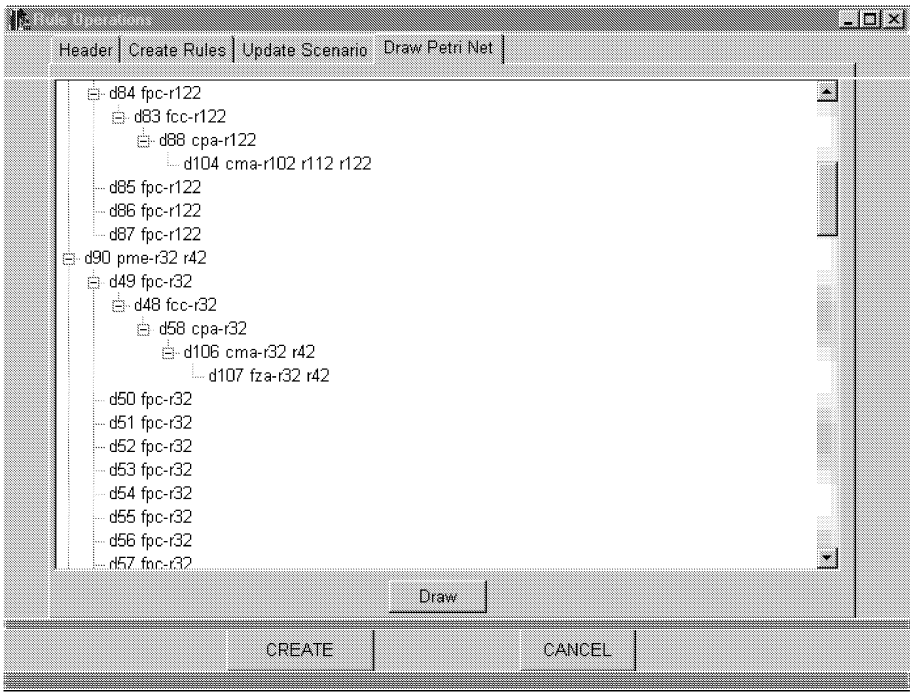
```
DEFINED RULES DOES NOT GUARANTEE CONFLUENCE:-(

RULE(s) r5 r7 r9 in GROUP 1 read forecast.weather in event part while RULE(s) r3 r4 in GROUP 0
write it.
```

Figure 7.8: Confluence Analysis for the example rules



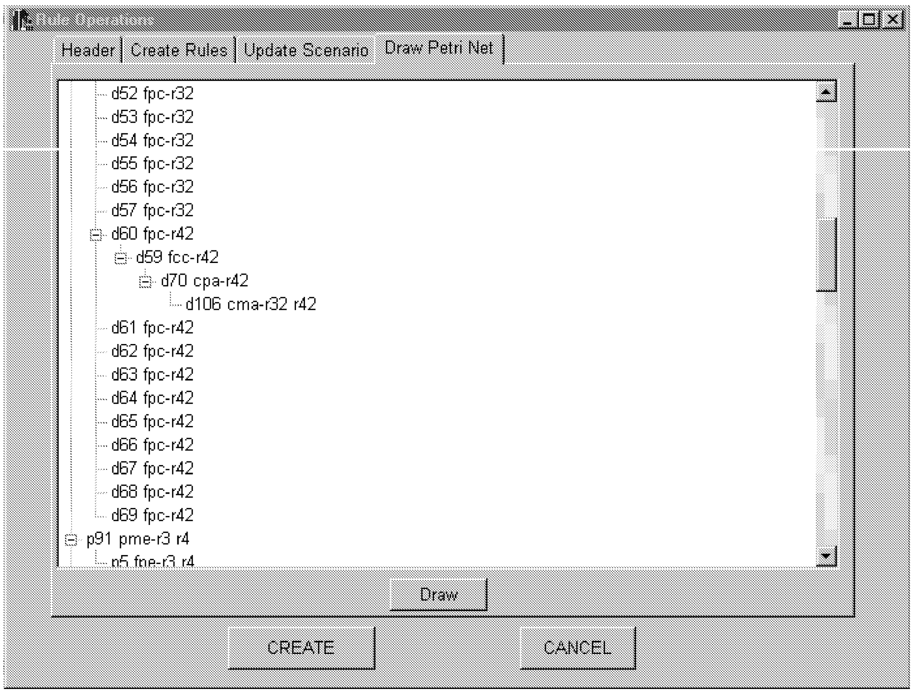
```
⊟ d89 pme-r102 r112 r122
   ⊟ d72 fpc-r102
      ⊟ d71 fcc-r102
         ⊟ d76 cpa-r102
            ⊟ d104 cma-r102 r112 r122
                 d105 fza-r102 r112 r122
     d73 fpc-r102
     d74 fpc-r102
     d75 fpc-r102
   ⊟ d78 fpc-r112
      ⊟ d77 fcc-r112
         ⊟ d82 cpa-r112
              d104 cma-r102 r112 r122
     d79 fpc-r112
     d80 fpc-r112
     d81 fpc-r112
   ⊟ d84 fpc-r122
      ⊟ d83 fcc-r122
         ⊟ d88 cpa-r122
              d104 cma-r102 r112 r122
     d85 fpc-r122
```

Figure 7.9: Petri Net (1)



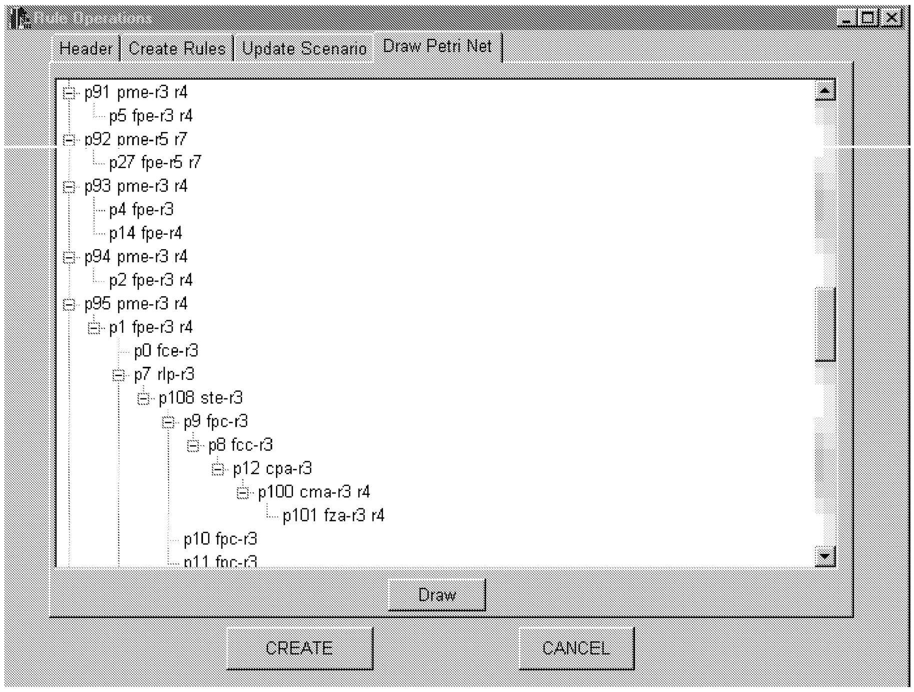
```
  d84 fpc-r122
    d83 fcc-r122
      d88 cpa-r122
        d104 cma-r102 r112 r122
  d85 fpc-r122
  d86 fpc-r122
  d87 fpc-r122
d90 pme-r32 r42
  d49 fpc-r32
    d48 fcc-r32
      d58 cpa-r32
        d106 cma-r32 r42
          d107 fza-r32 r42
  d50 fpc-r32
  d51 fpc-r32
  d52 fpc-r32
  d53 fpc-r32
  d54 fpc-r32
  d55 fpc-r32
  d56 fpc-r32
  d57 fpc-r32
```

Figure 7.10: Petri Net (2)



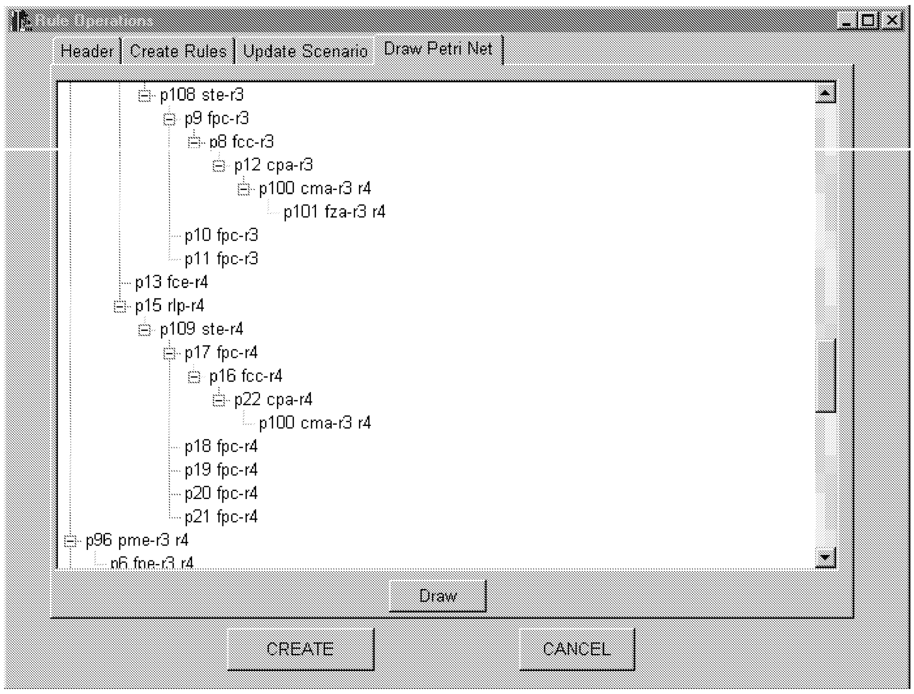
```
  d52 fpc-r32
  d53 fpc-r32
  d54 fpc-r32
  d55 fpc-r32
  d56 fpc-r32
  d57 fpc-r32
  d60 fpc-r42
    d59 fcc-r42
      d70 cpa-r42
        d106 cma-r32 r42
  d61 fpc-r42
  d62 fpc-r42
  d63 fpc-r42
  d64 fpc-r42
  d65 fpc-r42
  d66 fpc-r42
  d67 fpc-r42
  d68 fpc-r42
  d69 fpc-r42
p91 pme-r3 r4
  p5 fpe-r3 r4
```

Figure 7.11: Petri Net (3)



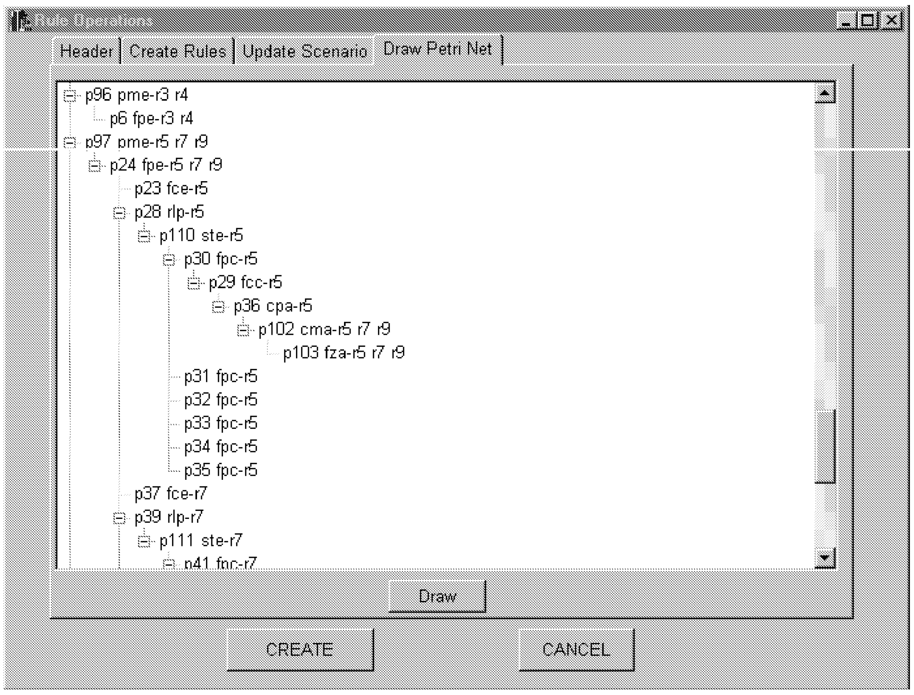
```
⊟ p91 pme-r3 r4
    p5 fpe-r3 r4
⊟ p92 pme-r5 r7
    p27 fpe-r5 r7
⊟ p93 pme-r3 r4
    p4 fpe-r3
    p14 fpe-r4
⊟ p94 pme-r3 r4
    p2 fpe-r3 r4
⊟ p95 pme-r3 r4
  ⊟ p1 fpe-r3 r4
      p0 fce-r3
    ⊟ p7 rlp-r3
      ⊟ p108 ste-r3
        ⊟ p9 fpc-r3
          ⊟ p8 fcc-r3
            ⊟ p12 cpa-r3
              ⊟ p100 cma-r3 r4
                  p101 fza-r3 r4
          p10 fpc-r3
          p11 fpc-r3
```

Figure 7.12: Petri Net (4)



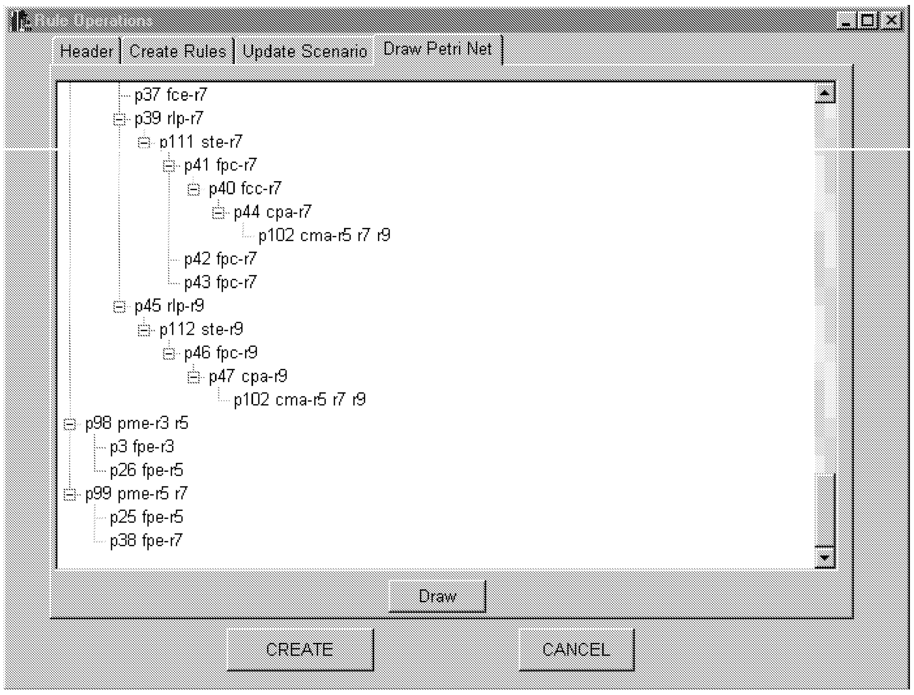
```
      ⊟ p108 ste-r3
        ⊟ p9 fpc-r3
          ⊟ p8 fcc-r3
            ⊟ p12 cpa-r3
              ⊟ p100 cma-r3 r4
                  p101 fza-r3 r4
          p10 fpc-r3
          p11 fpc-r3
      p13 fce-r4
    ⊟ p15 rlp-r4
      ⊟ p109 ste-r4
        ⊟ p17 fpc-r4
          ⊟ p16 fcc-r4
            ⊟ p22 cpa-r4
                p100 cma-r3 r4
          p18 fpc-r4
          p19 fpc-r4
          p20 fpc-r4
          p21 fpc-r4
⊟ p96 pme-r3 r4
    p6 fpe-r3 r4
```

Figure 7.13: Petri Net (5)



```
⊟ p96 pme-r3 r4
    p6 fpe-r3 r4
⊟ p97 pme-r5 r7 r9
    ⊟ p24 fpe-r5 r7 r9
        p23 fce-r5
        ⊟ p28 rlp-r5
            ⊟ p110 ste-r5
                ⊟ p30 fpc-r5
                    ⊟ p29 fcc-r5
                        ⊟ p36 cpa-r5
                            ⊟ p102 cma-r5 r7 r9
                                p103 fza-r5 r7 r9
                p31 fpc-r5
                p32 fpc-r5
                p33 fpc-r5
                p34 fpc-r5
                p35 fpc-r5
        p37 fce-r7
        ⊟ p39 rlp-r7
            ⊟ p111 ste-r7
                ⊟ p41 fpc-r7
```

Figure 7.14: Petri Net (6)



```
        p37 fce-r7
        ⊟ p39 rlp-r7
            ⊟ p111 ste-r7
                ⊟ p41 fpc-r7
                    ⊟ p40 fcc-r7
                        ⊟ p44 cpa-r7
                            p102 cma-r5 r7 r9
                p42 fpc-r7
                p43 fpc-r7
        ⊟ p45 rlp-r9
            ⊟ p112 ste-r9
                ⊟ p46 fpc-r9
                    ⊟ p47 cpa-r9
                        p102 cma-r5 r7 r9
⊟ p98 pme-r3 r5
    p3 fpe-r3
    p26 fpe-r5
⊟ p99 pme-r5 r7
    p25 fpe-r5
    p38 fpe-r7
```

Figure 7.15: Petri Net (7)

Table 7.3: Sensor Values at times t_1 & t_2 and their fuzzification results according to the linguistic terms used in the event/condition parts of the rules

Attribute	Sensor Values		Fuzzification Results for event / condition evaluation	
	at time t_1	at time t_2	R_3	R_4
Wind Value	6	6.5	0.5 / -	- / -
Cloud Orientation	0	135	0.5 / -	0.5 / -
Cloud Base	7200	1600	1 / -	1 / 0.5
Wind Direction	255	165	- / -	- / 0.66
Humidity	50	57	1 / -	1 / 0.5
Temperature value	-4.5	-1	1 / -	1 / 0.5
Pressure value	1003.5	1000	1 / -	1 / 0.5

7.6.1 Forecasting the Weather - Active Rule Example(1)

Suppose that we are in the Winter season and we have the sensor values for times t_1 and t_2 given at Table 7.3 for the weather related attributes of a city, where $t_1 < t_2$. We use the membership functions given in Appendix A for this weather forecasting application. The results after the fuzzification of these values considering the fuzzy linguistic terms taking place inside the event/condition parts of the rules are also listed in the table.

Steps of the Inference

1. *Fuzzify the event*

   Evaluate R_3:

   $\mu_{pressure\_change\_direction\_decreasing}(-3.5) = 1.$

   $\mu_{humidity\_change\_direction\_increasing}(+7) = 1.$

   $\mu_{wind\_value\_medium\_strong/strong}(6.5) = 0.5.$

   $\mu_{cloud\_orientation\_vertical}(135) = 0.5.$

   $\mu_{cloud\_base\_change\_direction\_decreasing}(-5600) = 1.$

   $\mu_{temperature\_change\_direction\_increasing}(+3.5) = 1.$

   min(1, 1, 0.5, 0.5, 1, 1) = 0.5. Since $0.5 > 0$, R_3 `gets fired`.

   Evaluate R_4:

$\mu_{pressure\_change\_direction\_decreasing}(-3.5) = 1.$

$\mu_{humidity\_change\_direction\_increasing}(+7) = 1.$

$\mu_{temperature\_change\_direction\_increasing}(+3.5) = 1.$

$\mu_{cloud\_base\_change\_direction\_decreasing}(-5600) = 1.$

$\mu_{cloud\_orientation\_horizontal}(135) = 0.5.$

min(1, 1, 1, 1, 0.5) = 0.5. Since 0.5 > 0, `R`$_4$ `gets fired`.

2. *Calculate the strength of event and check it with the rule threshold*

   For this $\mu_{winter}(r_i)$ values for the rules are calculated using the Formula 3.9.

   $\mu_{winter}(R_3)$=0, $\mu_{winter}(R_4)$=1

   and using the Equation 3.1, event strengths are calculated as:

   for R_3 ; 0 * 0.5 = 0.

   for R_4 ; 1 * 0.5 = 0.5

   Threshold values for the rules are given as 0.1, therefore only R_4 succeed this step.

3. *Find the Condition Match Factor*

   Evaluate R_4:

   $\mu_{pressure\_change\_velocity\_slow}(-3.5) = 0.5.$

   $\mu_{humidity\_change\_velocity\_slow}(+7) = 0.5.$

   $\mu_{temperature\_change\_velocity\_slow}(+3.5) = 0.5.$

   $\mu_{cloud\_base\_change\_velocity\_slow}(-5600) = 0.5.$

   $\mu_{wind\_direction\_southeast/south}(210) = 0.66.$

   So min(0.5, 0.5, 0.5, 0.5, 0.66) = 0.5. Since 0.5 > 0 R$_4$ passes this step.

4. *Find the clipping values*

   Upto this point only R$_4$ succeed. Its antecedent matching degree is 0.5 * 0.5 = 0.25.

   So clipping value of 0.25 is used for the action part of R$_4$, which is an forecasted weather of `cloudedstable`.

5. *Find the fuzzy action*

   Maximum of the clipping values are taken, since we have R$_4$, max(0.25) =

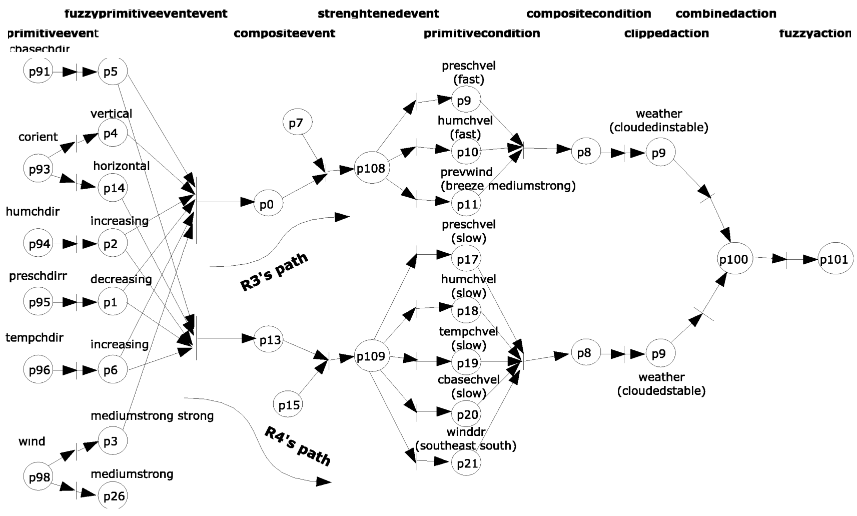


Figure 7.16: FPN used for forecasting the weather

0.25. This means a weather forecast of `cloudedstable` .

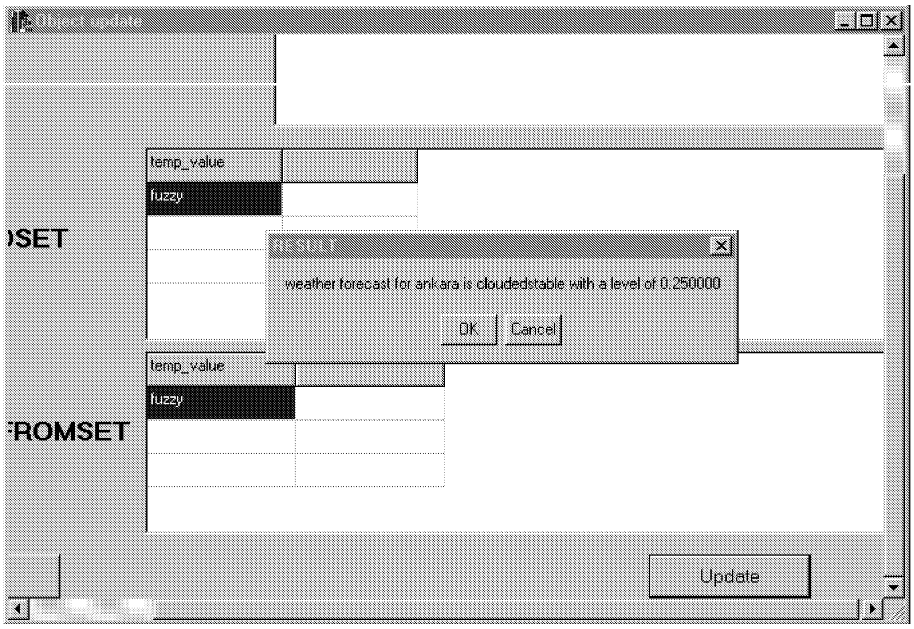
Figure 7.16 shows part of the FPN used for this inference. And, Figure 7.17 shows the screen which appears suddenly due to this dynamis(active) rule firing.

Section B.1 gives the application log for this inference. It shows movement of token values through the FPN.

7.6.2 Querying the Weather Forecast - Deductive Rule Example (1)

Suppose that after time t_2 a query comes like *X.forecast.weather*. We use the values given at Table 7.3 for evaluation of this query.
Note that, since deductive rules are fired, evaluation of this inference starts from conditions.

1. *Find the condition match factor*
   Evaluate R_{32}:
   min(1, 1, 0.5, 0.5, 1, 1, 0.5, 0.2, 1) = 0. $\mathtt{R}_{32}$ `fails`.
   Evaluate R_{42}:



weather forecast for ankara is cloudedstable with a level of 0.250000

Figure 7.17: Dynamic occurences (1)

min(1, 1, 1, 1, 0.5, 0.5,0.66,0.5,0.5) = 0.5. Since 0.5 > 0, R_4 `gets fired`.

2. *Find the clipping values*

   For this, the condition match factor is multiplied by the strength of event. For the abstract kind of events $\mu_{eventstrength}$ is always 1.

   for $R_{42}; 1 * 0.5 = 0.5$ (for the expected weather of `clouded stable`)

3. *Find the combined fuzzy action*

   We take the maximum of the clipped actions: $max(0.5) = 0.5$, which means expected weather of `clouded stable` is answered.

Figure 7.18 shows part of the FPN used for this example. Figure 7.19 shows the result of the query. B.2 gives the application log for this inference.

7.6.3 Forecasting the Weather Event - Active Rule Example (2)

Suppose that some more changes occur after this forecast as listed in the Table 7.4.

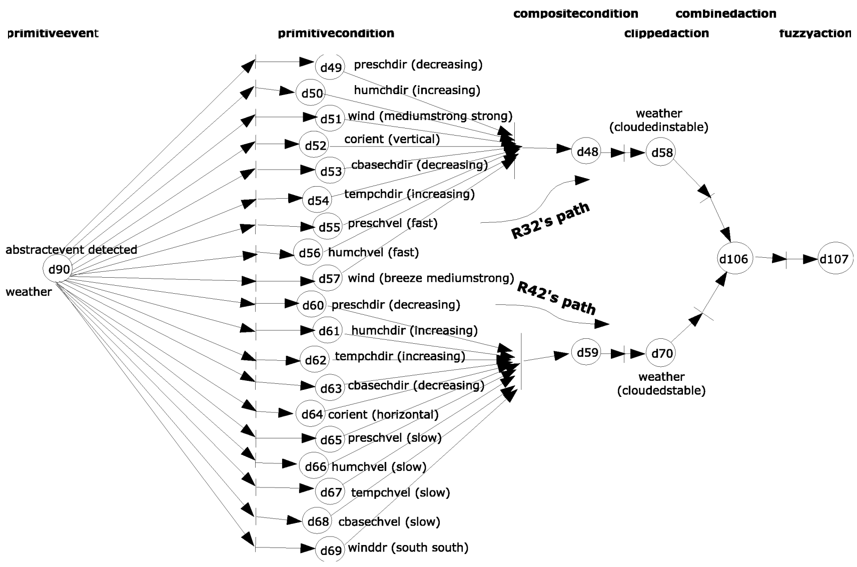


Figure 7.18: FPN used for querying the weather forecast

Steps of the Inference

1. *Fuzzify the event*

   Evaluate R_5:

   $\mu_{expected\_weather\_clouded\_stable} = 0.25$.

   $\mu_{wind\_direction\_south/southwest}(22.5) = 0$.

   $\mu_{wind\_value\_medium\_strong}(6) = 0$.

   $\mu_{cloud\_color\_grey}(4) = 0.5$.

   min(0.25,0,0,0.5) = 0, R_5 `fails`.

   Evaluate R_7:

   $\mu_{expected\_weather\_clouded\_stable} = 0.25$.

   $\mu_{wind\_direction\_north}(337.5) = 0.5$.

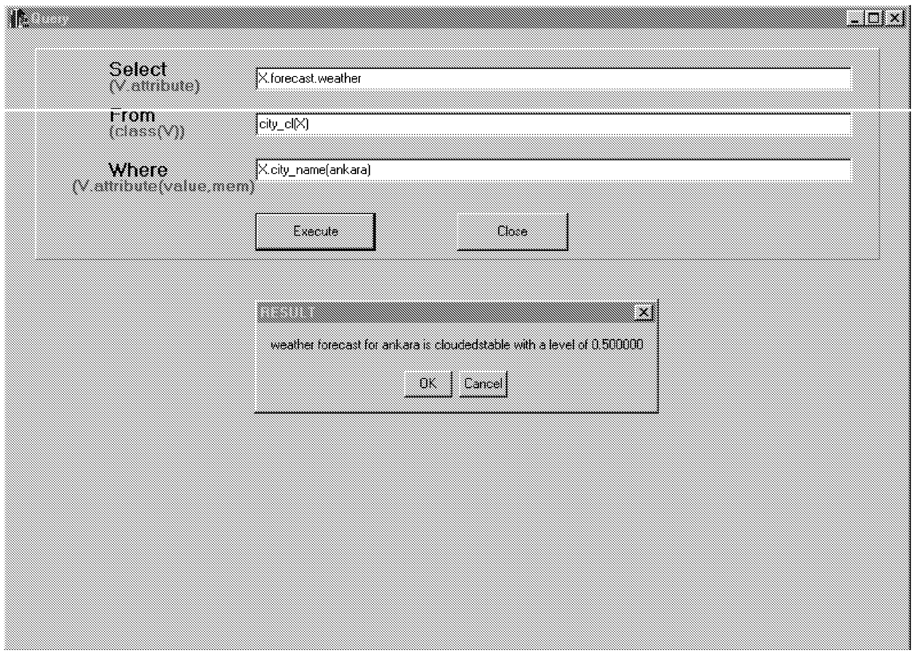
   $\mu_{cloud\_color\_grey}(4) = 0.5$.

   min(0.25, 0.5, 0.5) = 0.25. Since $0.25 > 0$ R_7 gets fired.

   Evaluate R_9:

   $\mu_{expected\_weather\_clouded\_stable} = 0.25$.

   min(0.25) = 0.25. Since $0.25 > 0$ R_9 `gets fired`.



X.forecast.weather

city_cl(X)

X.city_name(ankara)

weather forecast for ankara is cloudedstable with a level of 0.500000

Figure 7.19: A query requiring fuzzy inference on FPN (1)

2. *Calculate the strength of event and check it with the rule threshold*

   For this $\mu_{winter}(r_i)$ values for the rules are calculated using the Formula 3.9.

   $\mu_{winter}(R_7)$=1, $\mu_{winter}(R_9)$=1

   and using the Equation 3.1, event strengths are calculated as:

   for R_7 ; 1 * 0.25 = 0.25

   for R_9 ; 1 * 0.25 = 0.25

   Threshold values for the rules are given as 0.1, therefore R_7 and R_9 succeed the event_matching phase.

3. *Find the condition match factor*

   Evaluate R_7:

   $\mu_{temperature\_below\_0}(-1) = 1$.

   $\mu_{prev\_wind\_direction\_south}(210) = 0.66$.

   $\mu_{prev\_cloud\_color\_white}(3) = 0.5$.

   So min(1, 0.66, 0.5) = 0.5. Since 0.5 > 0, R$_7$ passes this step.

   Evaluate R_9:

   $\mu_{wind\_value\_calm/breeze}(6) = 1$.

Table 7.4: Sensor Values at times t_2 & t_3 ($t_2 < t_3$) and their fuzzification results according to the linguistic terms used in the event/condition parts of the rules

	Sensor Values		Fuzzification Results for event / condition evaluation		
Attribute	at time t_2	at time t_3	R_5	R_7	R_9
Weather	-	cloudedstable	0.25 / -	0.25 / -	0.25 / -
Wind Direction	165	22.5	0 / -	0.5 / 0.66	- / -
Wind Value	6.5	6	0 / -	- / -	- / 1
Cloud Color	3	4	0.5 / -	0.5 / 0.5	- / -
Pressure Value	1000	1003.5	- / -	- / 1	- / -

So min(1) = 1. Since $1 > 0$ R_9 passes this step.

4. *Find the clipping values*:

   Upto this point only R_7 and R_9 succeed. Their antecedent matching degrees are

   for R_7, 0.25 * 0.5 = 0.125.

   for R_9, 0.25* 1 = 0.25.

   So clipping value of 0.125 is used for the action part of R_7, which is an expected weather event of `snow` and clipping value of 0.25 is used for the action part of R_9, which is an expected weather event of `fog`.

5. *Find the fuzzy action*

   Maximum of the clipping values are taken, since we have R_7 and R_9, max(0.125,0.25) = 0.25. This means an expected weather event of `fog` is forecasted.

Figure 7.20 shows part of the FPN used for this inference. Figure 7.21 shows the screen for this execution. Section B.3 gives the application log for this inference.

7.6.4 Querying the Temperature Change Forecast - Deductive Rule Example (2)

Steps of the Inference

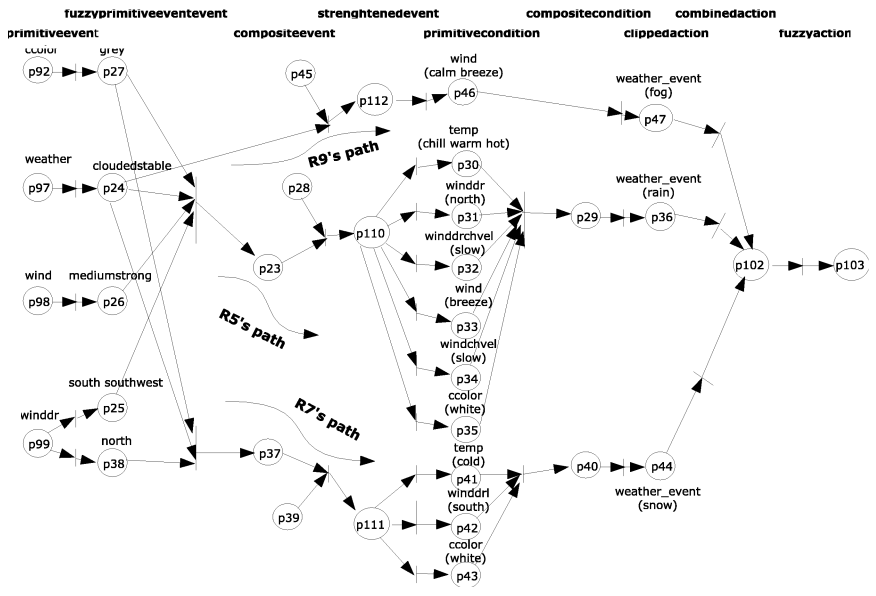


Figure 7.20: FPN constructed for forecasting the weather event

1. *Find the condition match factor*

   Evaluate R_{102}:

   $\mu_{expected\_weather\_clear/clear\_few/clouded\_instable}$ =0.5

   $\mu_{pressure\_change\_direction\_decreasing}(3.5) = 0.$

   $\mu_{wind\_direction\_south/southwest}(337.5) = 0.$

   $\mu_{prev\_wind\_direction\_north/northwest/northeast}(337.5) = 0.$

   min(0.5,0,0,0) = 0, $\mathtt{R}_{102}$ `fails`.

   Evaluate R_{112}:

   $\mu_{expected\_weather\_clouded\_stable} = 1.$

   $\mu_{pressure\_change\_direction\_increasing}(3.5) = 1.$

   $\mu_{wind\_direction\_north/northwest/northeast}(337.5) = 0.5.$

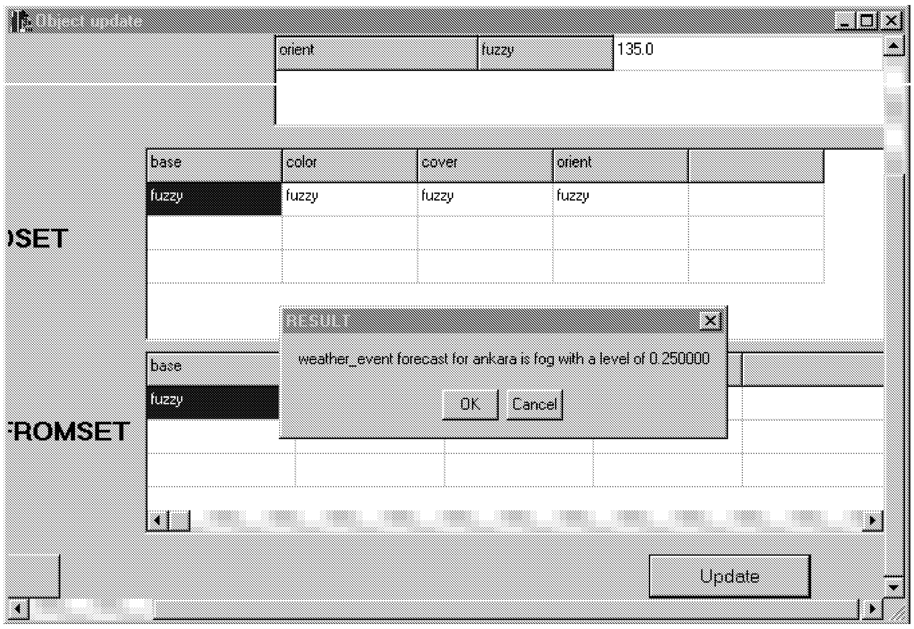
   $\mu_{prev\_wind\_direction\_south/southwest}(337.5) = 0.67.$

   min(1, 1, 0.5, 0.67) = 0.5. Since 0.5 > 0 $\mathtt{R}_{112}$ `gets fired`.

   Evaluate R_{122}:

   $\mu_{expected\_weather\_clear/clear\_few/clouded\_instable/clouded\_stable} = 1.$

   $\mu_{pressure\_change\_direction\_nochange}(3.5) = 0.$



weather_event forecast for ankara is fog with a level of 0.250000

Figure 7.21: Dynamic occurences (2)

$\mu_{wind\_direction\_west}(337.5) = 0.$

$\mu_{wind\_value\_breeze}(6) = 1.$

min(1,0,0,1) = 0. $\mathtt{R}_{122}$ `fails`.

2. *Find the clipping values*

   For this, the condition match factor is multiplied by the strength of event. For the abstract kind of events $\mu_{eventstrength}$ is always 1.

   for $R_{112}; 1 * 0.5 = 0.5$ (for the temperature change of `decrease`)

3. *Find the combined fuzzy action*

   We take the maximum of the clipped actions: $max(0.5) = 0.5$, which means a temperature change of `decrease` is answered.

Figure 7.22 shows part of the FPN used for this example. Figure 7.23 shows the result of this inference. Section B.4 gives the application log for this inference.

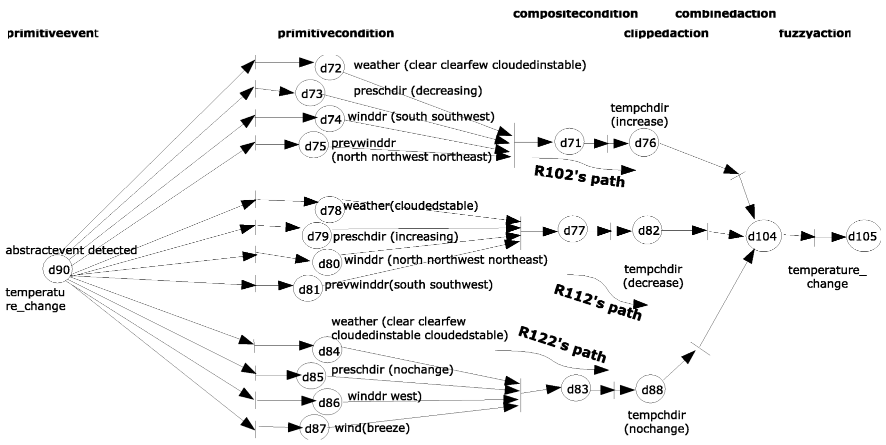
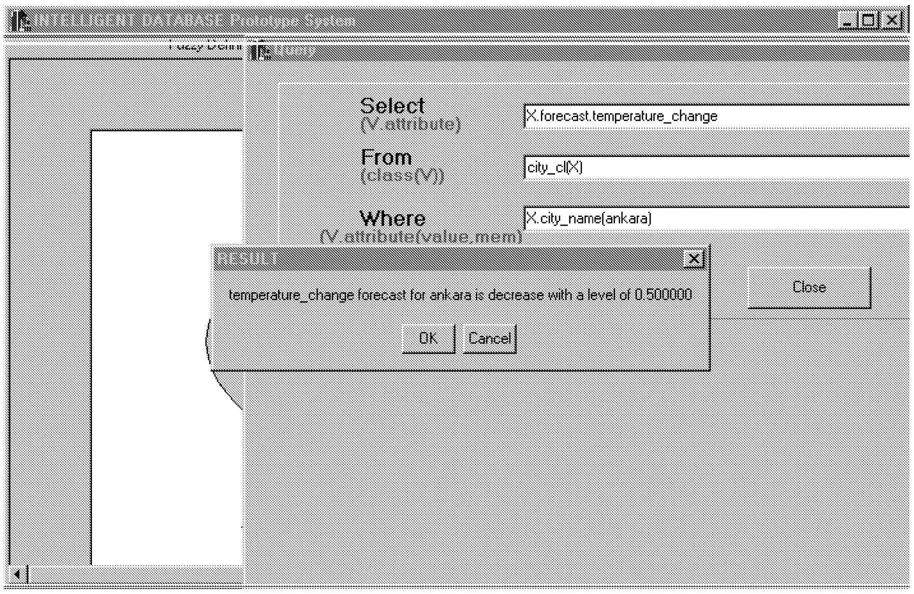


Figure 7.22: FPN used for querying the temperature change forecast



INTELLIGENT DATABASE Prototype System

Query

Select (V.attribute) X.forecast.temperature_change

From (class(V)) city_cl(X)

Where (V.attribute(value,mem)) X.city_name(ankara)

RESULT

temperature_change forecast for ankara is decrease with a level of 0.500000

OK Cancel

Close

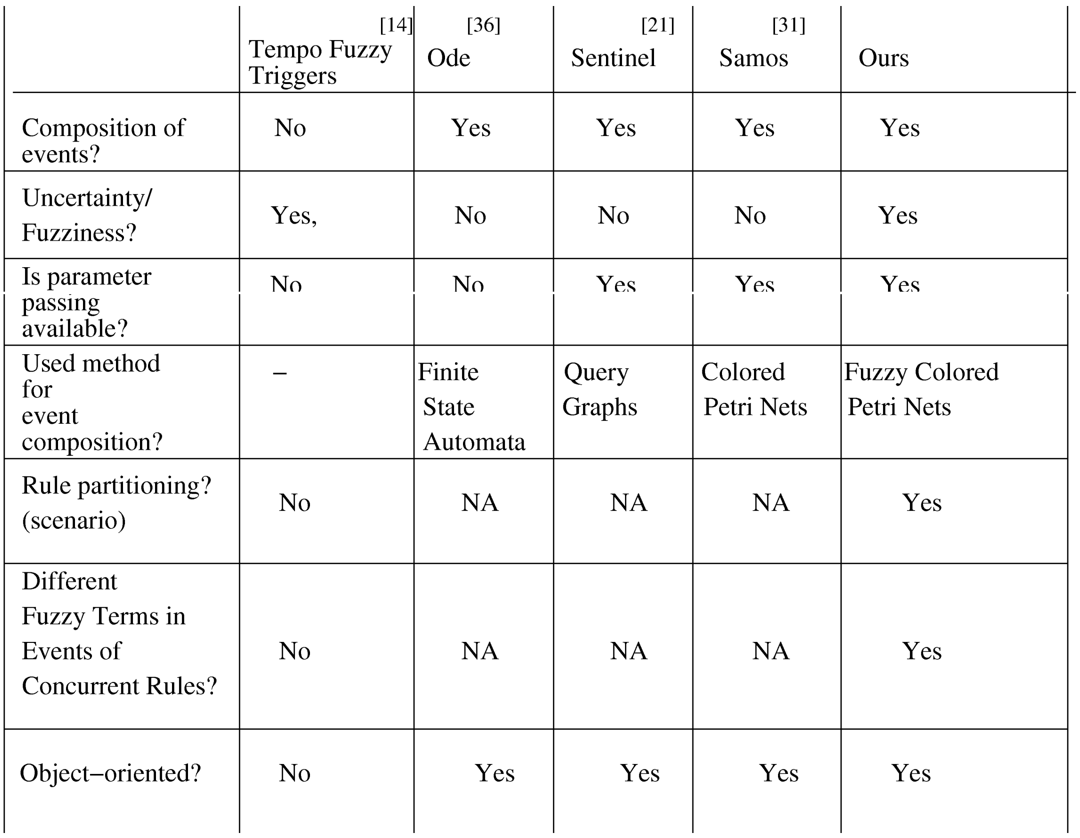
Figure 7.23: Query requiring Petri Net execution

CHAPTER 8

CONCLUSION AND FUTURE WORK

In this thesis, we introduce a FPN model for fuzzy rule-based reasoning. This includes the transformation of fuzzy rules into FPN, together with their reasoning. We can model active and deductive rules using the study introduced here. In addition, we can model all kinds of compositions (either event or condition) in our FPN. Besides, the functionalities of transitions are extended so that they are capable of performing some required computations in addition to the sup-min composition provided by other studies. Since we employ colored Petri nets, parameter passing is provided through the Petri net. This provides the values of conditions and actions to be calculated from the parameters of events.

By constructing the FPN, we inherently contain the triggering graph information. Therefore, without performing additional work, we can easily check the termination property of our system. When the assumptions and theories regarding the termination analysis were put forward, event and condition compositions were neglected by the previous studies [1, 3, 6, 19, 46]. On the other hand, we can handle event and condition compositions easily by the structures already provided by our FPN. In addition, our fuzzy reasoning algorithm working on our FPN assures confluent rule executions. This assurance comes from the fact that our reasoning algorithm provides a total order within the rules, using the similarity of the rules to the current active scenario.



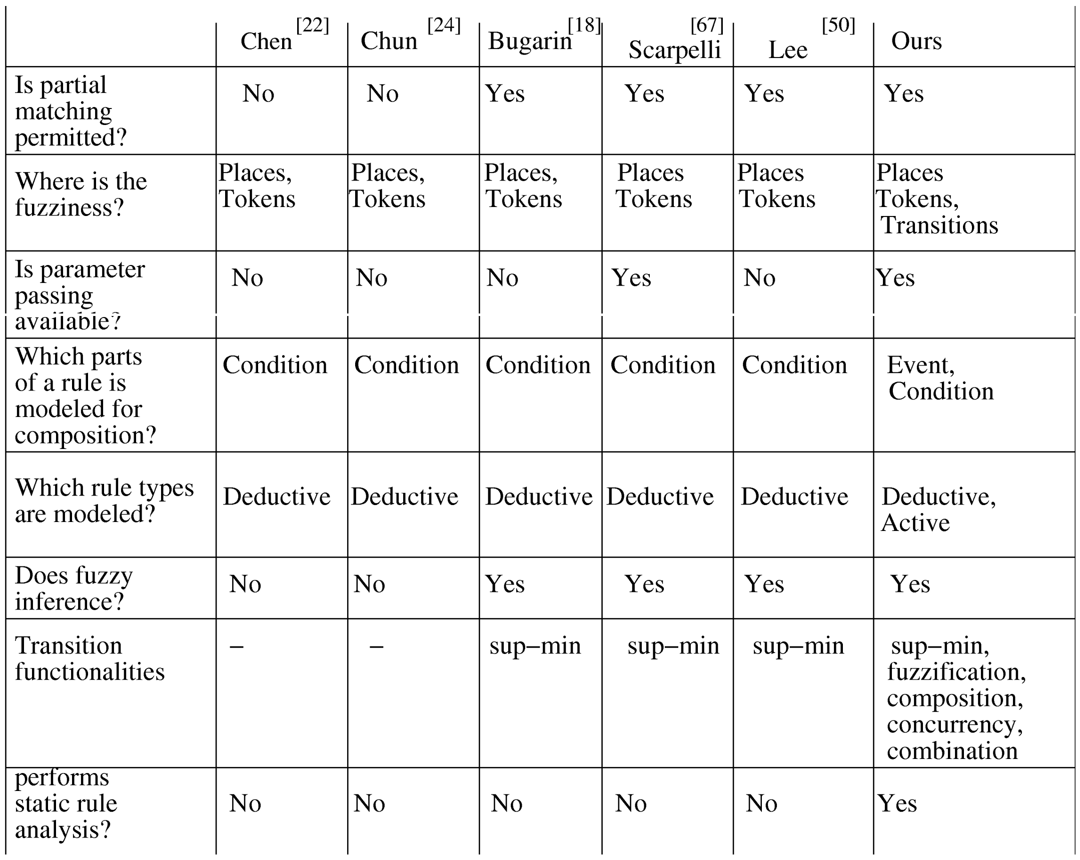
	[14] Tempo Fuzzy Triggers	[36] Ode	[21] Sentinel	[31] Samos	Ours
Composition of events?	No	Yes	Yes	Yes	Yes
Uncertainty/ Fuzziness?	Yes,	No	No	No	Yes
Is parameter passing available?	No	No	Yes	Yes	Yes
Used method for event composition?	–	Finite State Automata	Query Graphs	Colored Petri Nets	Fuzzy Colored Petri Nets
Rule partitioning? (scenario)	No	NA	NA	NA	Yes
Different Fuzzy Terms in Events of Concurrent Rules?	No	NA	NA	NA	Yes
Object–oriented?	No	Yes	Yes	Yes	Yes

Figure 8.1: Active Studies: A comparison.

Developing a full-fledged active object-oriented database environment is beyond the scope of this thesis. The point we consider is that, although there are a lot of studies on active databases, only few of them investigates fuzziness. On the other hand, the ones investigating fuzziness do not consider event/condition compositions. Figure 8.1 compares the previous active studies with ours.

There are also a couple of studies using FPNs. However, they do not model active rules. Also, they do not investigate the properties of their systems by using the FPNs. We aim to fill these gaps and propose a Fuzzy Petri net model to represent the knowledge and the behavior in an intelligent object-oriented database environment which integrates fuzzy, active and deductive rules with database objects. Figure 8.2 compares the previous studies using FPNs with ours.

Using the FPN model and the reasoning algorithm introduced in this study, a number of complex applications, such as weather forecasting applications, envi-



	Chen[22]	Chun[24]	Bugarin[18]	Scarpelli[67]	Lee[50]	Ours
Is partial matching permitted?	No	No	Yes	Yes	Yes	Yes
Where is the fuzziness?	Places, Tokens	Places, Tokens	Places, Tokens	Places Tokens	Places Tokens	Places Tokens, Transitions
Is parameter passing available?	No	No	No	Yes	No	Yes
Which parts of a rule is modeled for composition?	Condition	Condition	Condition	Condition	Condition	Event, Condition
Which rule types are modeled?	Deductive	Deductive	Deductive	Deductive	Deductive	Deductive, Active
Does fuzzy inference?	No	No	Yes	Yes	Yes	Yes
Transition functionalities	–	–	sup–min	sup–min	sup–min	sup–min, fuzzification, composition, concurrency, combination
performs static rule analysis?	No	No	No	No	No	Yes

Figure 8.2: Fuzzy Petri Nets: A comparison.

ronmental information systems, defense applications, workflow applications, etc., can be modeled. Furthermore, the techniques and solutions provided in this study for the static analysis of rule bases can be utilized to eliminate the undesirable behaviors of the modeled systems.

A prototype of the system is implemented by using ITASCA Object Database Management System and Borland C++ Builder. We have implemented the model using a weather forecasting application which uses the commonsense rules of the Central Anatolia Region.

Our work on Fuzzy Petri Nets can be extended in several ways. The followings are some research issues that can be studied as future work:

- In our current model, switching between the scenarios is determined by the user. However, a model can be developed for automatic switching between the scenarios.

- We assume all coupling modes being as immediate. Underlying aspects of the ITASCA DBMS (such as transaction manager) can be investigated whether it allows the imlementation of other coupling modes.
- Temporal dimension can be incorporated within the model.
- Performance of the FPN based inference can be improved via incorporating the RETE algorithm [33, 71] within the model.
- Validation of the weather forecasting rules that we defined here can be performed.
- The techniques developed here can be used for the implementation of video database applications.

REFERENCES

[1] Aiken A., Hellerstein J. and Widow J., "Static Analysis Techniques for predicting the Behavior of Active Database Rules", *ACM Transactions on Database Systems (ACM TODS'95)*, 20(1):3-41, March 1995.

[2] Al-Khatib W. and Ghafoor A., "An Approach for Video Meta-Data Modeling and Query Processing", in ACM Multimedia, pp.215-224, Orlando Florida, USA, 1999.

[3] Baralis E., Ceri S. and Paraboschi S., "Improved Rule Analysis by Means of Triggering and Activation Graphs", in Timos Sellis, editor, in *Proceedings of Second Workshop on Rules in Database Systems*, LNCS 985, pp.165-181, Greece, September 1995.

[4] Bancilhon F., Delobel C. and Kanellakis P., *Building an Object-Oriented Database System*, Morgan Kaufmann Publishers inc., 1992.

[5] Baralis E. "Rule Analysis", in Norman Paton, editor, *Active Rules in Database Systems*, pp.51-67, 1999.

[6] Baralis E. and Widow J., "An Algebraic Approach to Rule Analysis in Expert Database Systems", in *Proceedings of the 20th International Conference of Very Large Databases, (VLDB'94)*, pp.475-486, Chile, September 1994.

[7] Baralis E. and Widow J., "An Algebraic Approach to Static Analysis of Active Database Rules", in *ACM Transactions on Database Systems*, Vol.25, No.3, pp.269-332, 2000.

[8] Bardossy A., Duckstein L., and Bogardi I., "Fuzzy Rule-based Clssification of Atmospheric Circulation Patterns", *International Journal of Climatology*, Vol. 15, pp.1087-1097.

[9] Bayer P. and Jonker W., "A Framework for Supporting Triggers in Deductive Databases", in *Proceedings of 1st International workshop on Rules in Database Systems*, pp.316-330, Edinburg, Scotland, August 1993.

[10] Berndtsson M., "Reactive Object-Oriented Databases and CIM", in *Proceedings of the 5th International Conference on Database and Expert System Applications (DEXA'94)*, pp.769-778, Athens, Greece, September 1994.

[11] Bostan B. and Yazici A., "Fuzzy Inference Mechanism in Fuzzy ECA Rules", in *Proceedings of 10th International Fuzzy System Association World Congress, (IFSA'03)*, Istanbul, Turkey, June 29-July 2, 2003.

[12] Bostan-Korpeoglu B. and Yazici A., "An Active Object-Oriented Database Approach", in *Proceedings of 13th IEEE International Conference on Fuzzy Systems, (FUZZ-IEEE'04)*, Budapest, Hungary, 25-29 July, 2004.

[13] Bostan-Korpeoglu B. and Yazici A., "Using Fuzzy Petri Nets for Static Analysis of Rule-Bases", to appear in *Proceedings of 19th International Symposium on Computer and Information Sciences, (ISCIS'04)*, Antalya, Turkey, 27-29 October, 2004.

[14] Bouaziz T., Karvoven J., Pesonen A. and Wolski A., "Design and implementation of tempo fuzzy triggers", *Technical Report*, VTT Information Technology, 1997.

[15] Bouaziz T., Wolski A., "Incorporating Fuzzy Inference into Database Triggers", *Research Report*, VTT Information Technology, November 1996.

[16] Bouaziz T. and Wolski A., "Applying fuzzy events to approximate reasoning in active databases", in *Proceedings of 6th IEEE International Conference on Fuzzy Systems, (FUZZ-IEEE'97)*, Barcelona, Catalonia, Spain, July 1997.

[17] Buchmann A., "Active Object Systems", In Asuman Dogac, M. Tamer Ozsu, Alex Biliris, and Timos Sellis, editors, *Advances in Object-Oriented Database Systems*, pp.201-224, Spring-Verlag, 1994.

[18] Bugarin A. and Barro S., "Fuzzy Reasoning supported by Petri Nets", *IEEE Transactions on Fuzzy Systems*, 2(2), pp.135-150,1994.

[19] Ceri S. and Widow J., "Deriving Production Rules for Constraint Maintenance", in *Proceedings of the 16th International Conference of Very Large Databases, (VLDB'90)*, pp.566-577, Austria, August 1990.

[20] Ceri S. and Widom J., "Deriving Incremental Production Rules for Deductive Data", in *Information Systems*, Vol.16, No.6, 1994.

[21] Chakravarthy S., "SENTINEL: An Object-Oriented DBMS With Event-Based Rules", *SIGMOD Conference*, pp.572-575, 1997.

[22] Chen S., "Weighted Fuzzy Reasoning Using Weighted Fuzzy Petri Nets", *Transactions on Data Engineering, (TKDE'2002)*, Vol.14, No.2, March/April 2002.

[23] Chen S., Ke M., Chang J., "Knowledge Representation using Fuzzy Petri Nets", *Transactions on Data Engineering, (TKDE'90)*, 2(3), pp.311-319, 1990.

[24] Chun M. and Bien Z., "Fuzzy Petri Net Representation and Reasoning Methods for Rule-Based Decision Making Systems", *IECE Trans. Fundamentals*, Vol.E76-A, No.6, June 1993.

[25] Collet C., "NAOS", in *Active Rules in Database Systems*, pp.279-296, 1999.

[26] Ceri S., Fraternali P., Paraboschi S., Tanca L., "Active Rule Management in Chimera Active Database Systems", *Triggers and Rules For Advanced Database Processing* pp.151-176, 1996.

[27] Dayal U., "Active Database Management Systems", in *Proceedings of the 3rd International Conference on Data and Knowledge Bases*, pp.150-169, Jerusalem, June 1988.

[28] Dayal U., Buchmann A., Chakravarthy S., "The HiPAC Project Active Database Systems", in *Triggers and Rules For Advanced Database Processing*, pp.177-206, 1996.

[29] Diaz O., Jaime A., "EXACT: An Extensible Approach to Active Object-Oriented Databases", in *VLDB Journal*, 6(4), pp.282-295, 1997.

[30] Dinn A., Paton N. and Williams H., "Active Rule Analysis in the Rock and Roll Deductive Object-Oriented Database", *Information Systems*, Vol.24, No.4, pp.327-353, 1999.

[31] Dittrich K., Fritschi H., Gatziu S., Geppert A., Vaduva A., "SAMOS in hindsight: experiences in building an active object-oriented DBMS", in *Information Systems*, 28(5), pp.369-392, 2003.

[32] Elmasri R. and Navathe S.B., *Fundamentals of Database Systems*, The Benjamin/Cumming Publishing Com., 2000.

[33] Forgy C.L., "RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", in *Artificial Intelligence*, Vol.19, pp.17-37, 1982.

[34] Gatziu S. and Dittrich K., "Detecting Composite Events in Active Database Systems using Petri Nets", in *Proceedings of 4th International Workshop on Research issues in Data Engineering: Active Database Systems*, Houston, Texas, February 1994.

[35] Gatziu S., Geppert A., Dittrich K.R., Fritschi H., Vandau A., "Architecture and Implementation of the Active Object-Oriented Database Management System SAMOS", *Technical Report*, Institut fur Informatik, Universitat Zurich, October 1995.

[36] Gehani N., Jagadish H., "Active Database Facilities in Ode Active Database Systems", *Triggers and Rules For Advanced Database Processing*, pp.207-232, 1996.

[37] Gehani N. H. and Jagadish V., "Ode as an Active Database: Constraints and Triggers", in *Proceedings of 17th International Conference on Very Large Data Bases (VLDB'91)*, pp.327-336, Barcelona, Spain, September 1991.

[38] Ghanem N., DeMenthol D., Doermann D. and Davis L., "Representation and Recognition of Events in Surveillance Video Using Petri Nets", *Conference on Computer Vision and Pattern Recognition Workshop*, Vol.7, Washington D.C., USA.

[39] Goldsack P., *Weatherwise*, Newton Abbot, North Pomfret, Vermont, 1986.

[40] Hadjimicheal M., Kuciamuskas A., Tag P., "A meteorelogical fuzzy expert system incorporating subjective user input", *Knowledge and Information Systems*, Vol.4, No.3, pp.250-269, July 2002.

[41] Hansen B., "SIGMAR: A fuzzy expert system for critiquing marine forecasts", *AI Applications*, Vol.11, No.1, pp.59-68, 1997.

[42] Hansen B., "Analog Forecasting of Ceiling and visibility using fuzzy sets", *2nd Conference on Artificial Intelligence*, American Meteorological Society, pp.1-7, 2000.

[43] Harrington J.L., *Object-Oriented Database Design*, Morgan Kaufmann Publishers inc., 2000.

[44] Kwon O., Modeling and Generating Context-aware Agent-based Applications with Amended Colored Petri Nets, *Expert Systems with Applications* 27 (2004) 609-621.

[45] Kappel G. and Schrefl M., "Modeling Object Behaviour: To Use Methods or Rules or Both?", In Roland R. Wagner, Helmut Thoma (eds.): *Proceedings of the 7th International Conference on Database and Expert Systems Applications, (DEXA'96)*, Zurich, Switzerland, September 1996,, LNCS Vol.1134, pp.584-602, 1996.

[46] Karadimce A. and Urban S., "Refined Triggering Graphs: A Logic-Based Approach to Termination Analysis in an Object-Oriented Database", in *Proceedings of 12th International Conference on Data Engineering*, pp.384-391, New Orleans, February 1996.

[47] Kerry R., "Integrating Knowledge-Based and Database Management Systems", Ellis Horwood Limited, 1990.

[48] Kılıç A., "Herkes için Meteoroloji", *Doğa Araştırmaları ve Sporları Derneği (DASK)*, http://www.dask.org.tr/bilmek_istedikleriniz/meteoroloji/genel_sayfa.htm, May, 2004.

[49] Koyuncu K. and Yazici A., "IFOOD: An Intelligent Fuzzy Object-oriented Database Architecture", *IEEE Transactions on Knowledge and Data Engineering*, Vol.15, No.5, pp.1137-1154, 2003.

[50] Lee J., Liu K. and Chiang W., "Fuzzy Petri Nets for modelling rule-based reasoning", *International Journal on Artificial Intelligence Tools*, 1998.

[51] Lee J., Liu K. and Chiang W., "High Level Fuzzy Petri Nets as a basis for Managing Smbolic and Numerical information", *International Journal on Artificial Intelligence Tools*, Vol.9, No.4, pp.569-588, 2000.

[52] Lee J., Liu K. Wang Y. and Chiang W., "Possibilistic Petri Nets as a basis for agent service description language", *Fuzzy Sets and Systems*, 114, pp.105-126, 2004.

[53] Maner W., Joyse S., "WXSYS: Weather Lore + Fuzzy Logic = Weather Forecasts", presented in *CLIPS Virtual Conference*, 1997.

[54] Murata T., "Petri Nets: Properties, Analysis and Applications", in *Proceedings of IEEE*, Vol.77, pp.541-540, April, 1989.

[55] Manoj T., Leena J. and Soney R., "Knowledge Representation using Fuzzy Petri Nets- Revisited", *Transactions on Data Engineering, (TKDE'98)*, Vol.10, No.4, July/August, 1998.

[56] Montesi D., and Torlone R., "Analysis and Optimisation of Active Databases", *Data and Knowledge Engineering*, Vol.40, pp.241-271, 2002.

[57] Montesi D., Bagnato M. and Dallera C. "Termination Analysis in Active Databases", *International Database Engineering and Applications Symposium*, Montreal, 1999.

[58] Montesi D., Bertino E. and Bagnato M., "Refined Rules Termination Analysis through Transactions", *Information Systems*, Vol.28, pp.435-456, 2003.

[59] Motro A., "Accomodating Imprecision in Database Systems: Issue and Solutions", in *Sigmod Record*, Vol.19, No.4, pp.69-74, 1990.

[60] Motro A., "Imprecision and Uncertainty in Database Systems", in *Fuzziness in Database Management Systems* edited by P. Boch and J. Kacprzyk, Physica Verlag, pp.3-22, 1995.

[61] Murtha J., Applications of fuzzy Logic in operational meteorology, NewsLetter, Canadian Forces Weather Service, pp.42-54, 1995.

[62] Paton N., Diaz O., Williams M., Campin J., Dinn A. and Jaime A., "Dimensions of Active Behavior", in *Proceedings of 1st International Workshop on Rules in Database Systems*, pp.40-47, 1994.

[63] Paton N. and Diaz O., "Active Database Systems", *ACM Computing Surveys*, 31(1): pp.1-29, 1999.

[64] Pessonen A. and Wolski A., "Quantified and Temporal Fuzzy Reasoning for Active Monitoring in RapidBase", *Symposium on Tool Environments and Development Methods for Intelligent Systems (TOOLMET'2000)*, Finland, April 2000.

[65] Peterson J.L., "Petri Nets", *ACM Computing Surveys*, Vol.4, pp.223-252, 1977.

[66] Stonebraker M., "The Integration of Rule Systems and Database Systems", *IEEE Transactions on Knowledge and Data Engineering*, Vol.4, No.5, October, 1992.

[67] Scarpelli H., Gomide F. and Yager R., "A Reasoning Algorithm for High Level Fuzzy Petri Nets", *IEEE Transactions on Fuzzy Systems*, 4(3), pp.282-294, 1996.

[68] Saygin Y., Ulusoy O. and Yazici A., "Dealing with Fuzziness in Active Mobile Database Systems", *Information Sciences (International Journal)*, Vol.120, No.1-4, pp:23-44, December 1999.

[69] Shao J., Laux S.J., Trainor B.J., Pettifer R., "Nowcast of temperature and ice on overhead transmission wires", *Meteorological Applications*, Vo.10, pp.123-133, Cambridge University Press, 2003.

[70] Sujitjorn S., Sookjaras P., Wainikorn W., "An expert system to forecast visibility in Don-Muang Air Force Base", *IEEE International Conference on Systems, Man and Cybernetics*, pp.2528-2531, NY, USA, October 1994.

[71] Sosnowski Z.A., "Activation of Fuzzy Rules in RETE Network", in *Proceedings of the 4th International Conference on Flexible Query Answering Systems (FQAS'2000)*, Advances in Soft Computing, Physica-Verlag, pp.200-209, Warsaw, Poland, October 2000.

[72] Tag P., Hadjimicheal M., Brody L., Kuciauskas A. and Bankert R., "Automating the subjective recognition of wind Patterns as input to a meteorological forecasting system", in *Proceedings of 15th Conference on weather analysis and forecasting*, American Meteorological Society, pp.347-350, 1996.

[73] Tryfona N., "Modeling Phenomena in Spatiotemporal Dababases: Desiderata and solutions", in *Proceedings of 9th International Conference on Database and Expert Systems Applications, (DEXA'98)*, pp.155-165, 1998.

[74] Vaduva A., Gatziu S. and Dittrich K., "Investigating Termination in Active Database Systems with Expressive Rule Languages", *Technical Report*, Institut für Informatik, April 1997.

[75] Wolski A. and Bouaziz T., "Fuzzy Triggers: Incorporating Imprecise Reasoning into Active Databases", in *Proceedings of 14th International Conference on Data Engineering (ICDE'98)*, February 1998.

[76] Yayvan M., Weather Forecast Expert, in *Turkish State Meteorological Service*, Interview, May 2004.

[77] Yazici A., George R. and Aksoy D., "Design and Implementation Issues in the Fuzzy Object-oriented Data (FOOD) Model", *Information Sciences (Int. Journal)*, Vol.108, 1998.

[78] Yen J. and Langari R., *Fuzzy Logic, Intelligence, Control and Information*, New Jersey, USA, 1999.

[79] Zimmer D., Meckenstock A. and Unland R., "Using Petri Nets for Rule Termination Analysis", *Workshop on Databases: Active and Real-time*, Rockville, Maryland, November 1996.

[80] Zadeh L.A., "Fuzzy Sets", *Information and Control*, Vol.8, No.3, pp.338-353, 1965.

[81] Zaniolo C., "On the Unification of Active and Deductive Databases", in *Advances in Databases* 11th British National Conference on Databases, pp.23-39, Springer-Verlag LNCS 696, 1993.

[82] Zimmermann J., Buchmann A., "REACH", *Active Rules in Database Systems*, pp.263-277, 1999.

APPENDIX A

CLASS DEFINITIONS AND FUZZY DOMAINS OF METEOROLOGICAL DATA

Class Pressure{
attribute fuzzyPres value ;
attribute fuzzyPres prev_value ;
attribute fuzzyPreschdir change_direction ;
attribute fuzzyPreschvel change_velocity ;}

Class Temperature{
attribute fuzzyTemp value ;
attribute fuzzyTemp prev_value ;
attribute fuzzyTempchdir change_direction ;
attribute fuzzyTempchvel change_velocity ;}

Class Humidity{
attribute fuzzyHum value ;
attribute fuzzyHum prev_value ;
attribute fuzzyHumchdir change_direction ;
attribute fuzzyHumchvel change_velocity ;}

Class Wind{

attribute fuzzyWind value ;

attribute fuzzyWind prev_value ;

attribute fuzzyWindchvel change_velocity ;

attribute fuzzyWinddr direction ;

attribute fuzzyWinddr prev_direction;

attribute fuzzyWinddrchvel direction_change_velocity ;}

Class Cloud{

attribute fuzzyCcover cloud_cover;

attribute fuzzyCcover prev_cloud_cover;

attribute fuzzyCbase cloud_base;

attribute fuzzyCbase prev_cloud_base;

attribute fuzzyCbasechdir cloud_base_change_direction;

attribute fuzzyCbasechvel cloud_base_change_velocity;

attribute fuzzyOrient cloud_orientation;

attribute fuzzyColor cloud_color ;}

Class Forecast{

attribute fuzzyWeather expected_weather;

attribute fuzzyWeatherEvent expected_weather_event;

attribute fuzzyTemperatureChange expected_temperature_change;}

Class City{

attribute Pressure pressure;

attribute Temperature temperature;

attribute Humidity humidity;

attribute Wind wind;

attribute Cloud cloud;

attribute Forecast forecast;}

Table A.1: Membership Functions for *pressure value* attribute

fuzzy term	*function*	*arguments*			
low	trapezoidal	992	992	1008	1024
high	trapezoidal	1008	1024	1040	1040

Table A.2: Membership Functions for *pressure change velocity* attribute

fuzzy term	*function*	*arguments*			
nochange	trapezoidal	0	0	1	2
slow	trapezoidal	1	2	3	4
fast	trapezoidal	3	4	5	5

Table A.3: Membership Functions for *pressure change direction* attribute

fuzzy term	*function*	*arguments*			
decreasing	trapezoidal	-5	-5	-3	-1
nochange	trapezoidal	-3	-1	1	3
increasing	trapezoidal	1	3	5	5

Table A.4: Membership Functions for *temperature value* attribute

fuzzy term	*function*	*arguments*			
cold	trapezoidal	-5	-5	5	10
chill	trapezoidal	5	10	15	20
warm	trapezoidal	15	20	25	30
hot	trapezoidal	25	30	35	35

Table A.5: Membership Functions for *temperature change velocity* attribute

fuzzy term	*function*	*arguments*			
nochange	trapezoidal	0	0	1	2
slow	trapezoidal	1	2	3	4
fast	trapezoidal	3	4	5	5

Table A.6: Membership Functions for *temperature change direction* attribute

fuzzy term	*function*	*arguments*			
decreasing	trapezoidal	-5	-5	-3	-1
nochange	trapezoidal	-3	-1	1	3
increasing	trapezoidal	1	3	5	5

Table A.7: Membership Functions for *humidity value* attribute

fuzzy term	*function*	*arguments*			
low	trapezoidal	0	0	20	40
ideal	trapezoidal	20	40	60	80
high	trapezoidal	60	80	100	100

Table A.8: Membership Functions for *humidity change velocity* attribute

fuzzy term	*function*	*arguments*			
nochange	trapezoidal	0	0	2	4
slow	trapezoidal	2	4	6	8
fast	trapezoidal	6	8	10	10

Table A.9: Membership Functions for *humidity change direction* attribute

fuzzy term	*function*	*arguments*			
decreasing	trapezoidal	-10	-10	-6	-2
nochange	trapezoidal	-6	-2	2	6
increasing	trapezoidal	2	6	10	10

Table A.10: Membership Functions for *wind value* attribute

fuzzy term	*function*	*arguments*			
calm	trapezoidal	0	0	3	4
breeze	trapezoidal	3	4	6	7
medium_strong	trapezoidal	6	7	15	16
strong	trapezoidal	15	16	30	31
storm	trapezoidal	30	31	50	50

Table A.11: Membership Functions for *wind value change velocity* attribute

fuzzy term	*function*	*arguments*			
nochange	trapezoidal	0	0	2	4
slow	trapezoidal	2	4	6	8
fast	trapezoidal	6	8	10	10

Table A.12: Membership Functions for *wind direction* attribute

fuzzy term	*function*	*arguments*		
north	triangular	-45	0	45
northwest	triangular	0	45	90
west	triangular	45	90	135
southwest	triangular	90	135	180
south	triangular	135	180	225
southeast	triangular	180	225	270
east	triangular	225	270	315
northeast	triangular	270	315	360

Table A.13: Membership Functions for *wind direction change velocity* attribute

fuzzy term	*function*	*arguments*			
nochange	trapezoidal	0	0	18	36
slow	trapezoidal	18	36	54	72
fast	trapezoidal	54	72	90	90

Table A.14: Membership Functions for *cloud cover* attribute

fuzzy term	*function*	*arguments*			
clear	triangular	0	0	1	-
few	trapezoidal	0	1	2	3
broken_sky	trapezoidal	2	3	4	5
cloudy	trapezoidal	4	5	6	7
overcast	trapezoidal	6	7	8	8

Table A.15: Membership Functions for *cloud base* attribute

fuzzy term	*function*	*arguments*			
low	trapezoidal	0	0	2000	3000
medium	trapezoidal	2000	3000	5000	6000
high	trapezoidal	5000	6000	8000	8000

Table A.16: Membership Functions for *cloud base change direction* attribute

fuzzy term	*function*	*arguments*			
decreasing	trapezoidal	-8000	-8000	-4800	-1600
nochange	trapezoidal	-4800	-1600	1600	4800
increasing	trapezoidal	1600	4800	8000	8000

Table A.17: Membership Functions for *cloud base change velocity* attribute

fuzzy term	*function*	*arguments*			
nochange	trapezoidal	0	0	1600	3200
slow	trapezoidal	1600	3200	4800	6400
fast	trapezoidal	4800	6400	8000	8000

Table A.18: Membership Functions for *cloud orientation* attribute

fuzzy term	*function*	*arguments*		
vertical	triangular	45	90	145
horizontal	triangular	-45	0	45

APPENDIX B

APPLICATION LOG FOR THE EXAMPLES

B.1 Forecasting the Weather - Active Rule Example(1)

```
 INFERENCE GROUP 0 FIRED FOR EXECUTION


CURRENT_PLACES FOR USE: p91 p93 p94 p95 p96 p98
PLACE FOR USE NOW: p91, TRANSITION FOR USE NOW: 0
I am in TRANSITION 0 now, which does FUZZIFICATION.
I mark the OUTPUT place p5, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p5:
   Object id: c5, Object attr value: -5600.000000, Domain: fuzzycbasechdir,
   Fuzzytermlist: decreasing, Membership: 1.00

CURRENT_PLACES FOR USE: p93 p94 p95 p96 p98 p5
PLACE FOR USE NOW: p93, TRANSITION FOR USE NOW: 2
I am in TRANSITION 2 now, which does FUZZIFICATION.
I mark the OUTPUT place p4, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p4:
   Object id: c5, Object attr value: 135.000000, Domain: fuzzycorient,
   Fuzzytermlist: vertical, Membership: 0.50

CURRENT_PLACES FOR USE: p93 p94 p95 p96 p98 p5 p4
PLACE FOR USE NOW: p93, TRANSITION FOR USE NOW: 3
```

```
I am in TRANSITION 3 now, which does FUZZIFICATION.
I mark the OUTPUT place p14, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p14:
   Object id: c5, Object attr value: 135.000000, Domain: fuzzycorient,
   Fuzzytermlist: horizontal, Membership: 0.50

CURRENT_PLACES FOR USE: p94 p95 p96 p98 p5 p4 p14
PLACE FOR USE NOW: p94, TRANSITION FOR USE NOW: 4
I am in TRANSITION 4 now, which does FUZZIFICATION.
I mark the OUTPUT place p2, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p2:
   Object id: c5, Object attr value: 7.000000, Domain: fuzzyhumchdir,
   Fuzzytermlist: increasing, Membership: 1.00

CURRENT_PLACES FOR USE: p95 p96 p98 p5 p4 p14 p2
PLACE FOR USE NOW: p95, TRANSITION FOR USE NOW: 5
I am in TRANSITION 5 now, which does FUZZIFICATION.
I mark the OUTPUT place p1, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p1:
   Object id: c5, Object attr value: -3.500000, Domain: fuzzypreschdir,
   Fuzzytermlist: decreasing, Membership: 1.00

CURRENT_PLACES FOR USE: p96 p98 p5 p4 p14 p2 p1
PLACE FOR USE NOW: p96, TRANSITION FOR USE NOW: 6
I am in TRANSITION 6 now, which does FUZZIFICATION.
I mark the OUTPUT place p6, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p6:
   Object id: c5, Object attr value: 3.500000, Domain: fuzzytempchdir,
   Fuzzytermlist: increasing, Membership: 1.00

CURRENT_PLACES FOR USE: p98 p5 p4 p14 p2 p1 p6
PLACE FOR USE NOW: p98, TRANSITION FOR USE NOW: 8
I am in TRANSITION 8 now, which does FUZZIFICATION.
```

```
I mark the OUTPUT place p3, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p3:
   Object id: c5, Object attr value: 6.500000, Domain: fuzzywind,
   Fuzzytermlist: mediumstrong strong, Membership: 0.50

CURRENT_PLACES FOR USE: p98 p5 p4 p14 p2 p1 p6 p3
PLACE FOR USE NOW: p98, TRANSITION FOR USE NOW: 9
I am in TRANSITION 9 now, which does FUZZIFICATION.
Since TRANSITION 9 DOES NOT WORK for INFERENCE GROUP 0, I am not using it.
CURRENT_PLACES FOR USE: p5 p4 p14 p2 p1 p6 p3
PLACE FOR USE NOW: p5, TRANSITION FOR USE NOW: 43
I am in TRANSITION 43 now, which does COMPOSITION.
I mark the OUTPUT place p0, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p0:
   Object id: c5, Membership1: 0.50, Membership2: 0.00

CURRENT_PLACES FOR USE: p5 p4 p14 p2 p1 p6 p3 p0
PLACE FOR USE NOW: p5, TRANSITION FOR USE NOW: 44
I am in TRANSITION 44 now, which does COMPOSITION.
I mark the OUTPUT place p13, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p13:
   Object id: c5, Membership1: 0.50, Membership2: 0.00

CURRENT_PLACES FOR USE: p4 p14 p2 p1 p6 p3 p0 p13
PLACE FOR USE NOW: p4, TRANSITION FOR USE NOW: 43
Since transition 43 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: p14 p2 p1 p6 p3 p0 p13
PLACE FOR USE NOW: p14, TRANSITION FOR USE NOW: 44
Since transition 44 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: p2 p1 p6 p3 p0 p13
PLACE FOR USE NOW: p2, TRANSITION FOR USE NOW: 43
Since transition 43 is consumed before, do not fire it again.
```

```
CURRENT_PLACES FOR USE: p2 p1 p6 p3 p0 p13
PLACE FOR USE NOW: p2, TRANSITION FOR USE NOW: 44
Since transition 44 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: p1 p6 p3 p0 p13
PLACE FOR USE NOW: p1, TRANSITION FOR USE NOW: 43
Since transition 43 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: p1 p6 p3 p0 p13
PLACE FOR USE NOW: p1, TRANSITION FOR USE NOW: 44
Since transition 44 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: p6 p3 p0 p13
PLACE FOR USE NOW: p6, TRANSITION FOR USE NOW: 43
Since transition 43 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: p6 p3 p0 p13
PLACE FOR USE NOW: p6, TRANSITION FOR USE NOW: 44
Since transition 44 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: p3 p0 p13
PLACE FOR USE NOW: p3, TRANSITION FOR USE NOW: 43
Since transition 43 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: p0 p13
PLACE FOR USE NOW: p0, TRANSITION FOR USE NOW: 56
I am in TRANSITION 56 now, which does STRENGTH calculation.
Since strength result is 0, rule r3 is removed from the inference.

CURRENT_PLACES FOR USE: p13
PLACE FOR USE NOW: p13, TRANSITION FOR USE NOW: 61
I am in TRANSITION 61 now, which does STRENGTH calculation.
I mark the OUTPUT place p109, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p109:
   Object id: c5, Membership1: 0.50.
```

```
CURRENT_PLACES FOR USE: p109
PLACE FOR USE NOW: p109, TRANSITION FOR USE NOW: 62
I am in TRANSITION 62 now, which does TRIGGERING condition.
I mark the OUTPUT place p17, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p17:
   Object id: c5, Object attr value: 3.500000, Domain: fuzzypreschvel,
   Fuzzytermlist: slow, Membership1: 0.50, Membership2: 0.50

CURRENT_PLACES FOR USE: p109 p17
PLACE FOR USE NOW: p109, TRANSITION FOR USE NOW: 63
I am in TRANSITION 63 now, which does TRIGGERING condition.
I mark the OUTPUT place p18, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p18:
   Object id: c5, Object attr value: 7.000000, Domain: fuzzyhumchvel,
   Fuzzytermlist: slow, Membership1: 0.50, Membership2: 0.50

CURRENT_PLACES FOR USE: p109 p17 p18
PLACE FOR USE NOW: p109, TRANSITION FOR USE NOW: 64
I am in TRANSITION 64 now, which does TRIGGERING condition.
I mark the OUTPUT place p19, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p19:
   Object id: c5, Object attr value: 3.500000, Domain: fuzzyhumchvel,
   Fuzzytermlist: slow, Membership1: 0.75, Membership2: 0.50

CURRENT_PLACES FOR USE: p109 p17 p18 p19
PLACE FOR USE NOW: p109, TRANSITION FOR USE NOW: 65
I am in TRANSITION 65 now, which does TRIGGERING condition.
I mark the OUTPUT place p20, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p20:
   Object id: c5, Object attr value: 5600.000000, Domain: fuzzycbasechvel,
   Fuzzytermlist: slow, Membership1: 0.50, Membership2: 0.50
```

```
CURRENT_PLACES FOR USE: p109 p17 p18 p19 p20

PLACE FOR USE NOW: p109, TRANSITION FOR USE NOW: 66

I am in TRANSITION 66 now, which does TRIGGERING condition.

I mark the OUTPUT place p21, which is added to the tail of

CURRENT places list

TOKEN VALUE put in OUTPUT place p21:

   Object id: c5, Object attr value: 165.000000, Domain: fuzzywinddr,

   Fuzzytermlist: southeast south, Membership1: 0.67, Membership2: 0.50


CURRENT_PLACES FOR USE: p17 p18 p19 p20 p21

PLACE FOR USE NOW: p17, TRANSITION FOR USE NOW: 48

I am in TRANSITION 48 now, which does COMPOSITION.

I mark the OUTPUT place p16, which is added to the tail of

CURRENT places list

TOKEN VALUE put in OUTPUT place p16:

   Object id: c5, Membership1: 0.50, Membership2: 0.50


CURRENT_PLACES FOR USE: p18 p19 p20 p21 p16

PLACE FOR USE NOW: p18, TRANSITION FOR USE NOW: 48

Since transition 48 is consumed before, do not fire it again.


CURRENT_PLACES FOR USE: p19 p20 p21 p16

PLACE FOR USE NOW: p19, TRANSITION FOR USE NOW: 48

Since transition 48 is consumed before, do not fire it again.


CURRENT_PLACES FOR USE: p20 p21 p16

PLACE FOR USE NOW: p20, TRANSITION FOR USE NOW: 48

Since transition 48 is consumed before, do not fire it again.


CURRENT_PLACES FOR USE: p21 p16

PLACE FOR USE NOW: p21, TRANSITION FOR USE NOW: 48

Since transition 48 is consumed before, do not fire it again.


CURRENT_PLACES FOR USE: p16

PLACE FOR USE NOW: p16, TRANSITION FOR USE NOW: 67

I am in TRANSITION 67 now, which does CLIPPING.

I mark the OUTPUT place p22, which is added to the tail of
```

```
CURRENT places list
TOKEN VALUE put in OUTPUT place p22:
   Object id: c5, Membership1: 0.25

CURRENT_PLACES FOR USE: p22
PLACE FOR USE NOW: p22, TRANSITION FOR USE NOW: 90
I am in TRANSITION 90 now, which does COMBINATION.
I mark the OUTPUT place p100, which is added to the tail of
CURRENT places list
CLIPPED TOKEN VALUE put in OUTPUT place p100:
   Object id: c5, Fuzzytermlist: cloudedstable, Clipping value: 0.25

CURRENT_PLACES FOR USE: p100
PLACE FOR USE NOW: p100, TRANSITION FOR USE NOW: 91
I am in TRANSITION 91 now, which finds the overall FUZZY ACTION.
I mark the OUTPUT place p101, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p101:
   Object id: c5, Fuzzytermlist: cloudedstable, Overall Clipping value: 0.25

CURRENT_PLACES FOR USE: p101
PLACE FOR USE NOW: p101, TRANSITION FOR USE NOW: 103
I am in TRANSITION 103 now, which GENERATES EVENT.
I mark the OUTPUT place p97, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p97:
   Object id: c5, Membership: 0.25
```

B.2 Querying the Weather Forecast - Deductive Rule Example (1)

```
 INFERENCE GROUP 3 FIRED FOR EXECUTION

CURRENT_PLACES FOR USE: d90
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 24
I am in TRANSITION 24 now, which does TRIGGERING condition.
I mark the OUTPUT place d49, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d49:
   Object id: c5, Object attr value: -3.500000, Domain: fuzzypreschdir,
   Fuzzytermlist: decreasing, Membership1: 1.00, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 25
I am in TRANSITION 25 now, which does TRIGGERING condition.
I mark the OUTPUT place d50, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d50:
   Object id: c5, Object attr value: 7.000000, Domain: fuzzyhumchdir,
   Fuzzytermlist: increasing, Membership1: 1.00, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49 d50
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 26
I am in TRANSITION 26 now, which does TRIGGERING condition.
I mark the OUTPUT place d51, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d51:
   Object id: c5, Object attr value: 6.500000, Domain: fuzzywind,
   Fuzzytermlist: mediumstrong strong, Membership1: 0.50, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49 d50 d51
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 27
I am in TRANSITION 27 now, which does TRIGGERING condition.
I mark the OUTPUT place d52, which is added to the tail of
CURRENT places list
```

```
TOKEN VALUE put in OUTPUT place d52:
   Object id: c5, Object attr value: 135.000000, Domain: fuzzycorient,
   Fuzzytermlist: vertical, Membership1: 0.50, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49 d50 d51 d52
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 28
I am in TRANSITION 28 now, which does TRIGGERING condition.
I mark the OUTPUT place d53, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d53:
   Object id: c5, Object attr value: -5600.000000, Domain: fuzzycbasechdir,
   Fuzzytermlist: decreasing, Membership1: 1.00, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49 d50 d51 d52 d53
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 29
I am in TRANSITION 29 now, which does TRIGGERING condition.
I mark the OUTPUT place d54, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d54:
   Object id: c5, Object attr value: 3.500000, Domain: fuzzytempchdir,
   Fuzzytermlist: increasing, Membership1: 1.00, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49 d50 d51 d52 d53 d54
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 30
I am in TRANSITION 30 now, which does TRIGGERING condition.
I mark the OUTPUT place d55, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d55:
   Object id: c5, Object attr value: 3.500000, Domain: fuzzypreschvel,
   Fuzzytermlist: fast, Membership1: 0.50, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49 d50 d51 d52 d53 d54 d55
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 31
I am in TRANSITION 31 now, which does TRIGGERING condition.
I mark the OUTPUT place d56, which is added to the tail of CURRENT places list
TOKEN VALUE put in OUTPUT place d56:
   Object id: c5, Object attr value: 7.000000, Domain: fuzzyhumchvel,
   Fuzzytermlist: fast, Membership1: 0.50, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49 d50 d51 d52 d53 d54 d55 d56
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 32
I am in TRANSITION 32 now, which does TRIGGERING condition.
I mark the OUTPUT place d57, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d57:
   Object id: c5, Object attr value: 6.000000, Domain: fuzzywind,
   Fuzzytermlist: breeze mediumstrong, Membership1: 0.00, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49 d50 d51 d52 d53 d54 d55 d56 d57
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 33
I am in TRANSITION 33 now, which does TRIGGERING condition.
I mark the OUTPUT place d60, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d60:
   Object id: c5, Object attr value: -3.500000, Domain: fuzzypreschdir,
   Fuzzytermlist: decreasing, Membership1: 1.00, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49 d50 d51 d52 d53 d54 d55 d56 d57 d60
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 34
I am in TRANSITION 34 now, which does TRIGGERING condition.
I mark the OUTPUT place d61, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d61:
   Object id: c5, Object attr value: 7.000000, Domain: fuzzyhumchdir,
   Fuzzytermlist: increasing, Membership1: 1.00, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49 d50 d51 d52 d53 d54 d55 d56 d57 d60 d61
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 35
I am in TRANSITION 35 now, which does TRIGGERING condition.
I mark the OUTPUT place d62, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d62:
   Object id: c5, Object attr value: 3.500000, Domain: fuzzytempchdir,
   Fuzzytermlist: increasing, Membership1: 1.00, Membership2: 1.00
```

```
CURRENT_PLACES FOR USE: d90 d49 d50 d51 d52 d53 d54 d55 d56 d57 d60 d61 d62
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 36
I am in TRANSITION 36 now, which does TRIGGERING condition.
I mark the OUTPUT place d63, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d63:
   Object id: c5, Object attr value: -5600.000000, Domain: fuzzycbasechdir,
   Fuzzytermlist: decreasing, Membership1: 1.00, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49 d50 d51 d52 d53 d54 d55 d56 d57 d60 d61 d62 d63
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 37
I am in TRANSITION 37 now, which does TRIGGERING condition.
I mark the OUTPUT place d64, which is added to the tail of CURRENT places list
TOKEN VALUE put in OUTPUT place d64:
   Object id: c5, Object attr value: 135.000000, Domain: fuzzycorient,
   Fuzzytermlist: horizontal, Membership1: 0.50, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49 d50 d51 d52 d53 d54 d55 d56 d57 d60 d61 d62 d63 d64
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 38
I am in TRANSITION 38 now, which does TRIGGERING condition.
I mark the OUTPUT place d65, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d65:
   Object id: c5, Object attr value: 3.500000, Domain: fuzzypreschvel,
   Fuzzytermlist: slow, Membership1: 0.50, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49 d50 d51 d52 d53 d54 d55 d56 d57 d60 d61 d62 d63 d64 d65
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 39
I am in TRANSITION 39 now, which does TRIGGERING condition.
I mark the OUTPUT place d66, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d66:
   Object id: c5, Object attr value: 7.000000, Domain: fuzzyhumchvel,
   Fuzzytermlist: slow, Membership1: 0.50, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49 d50 d51 d52 d53 d54 d55 d56 d57 d60 d61 d62 d63
```

```
d64 d65 d66
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 40
I am in TRANSITION 40 now, which does TRIGGERING condition.
I mark the OUTPUT place d67, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d67:
   Object id: c5, Object attr value: 3.500000, Domain: fuzzytempchvel,
   Fuzzytermlist: slow, Membership1: 0.50, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49 d50 d51 d52 d53 d54 d55 d56 d57 d60 d61 d62 d63
d64 d65 d66 d67
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 41
I am in TRANSITION 41 now, which does TRIGGERING condition.
I mark the OUTPUT place d68, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d68:
   Object id: c5, Object attr value: 5600.000000, Domain: fuzzycbasechvel,
   Fuzzytermlist: slow, Membership1: 0.50, Membership2: 1.00

CURRENT_PLACES FOR USE: d90 d49 d50 d51 d52 d53 d54 d55 d56 d57 d60 d61 d62 d63 d64
d65 d66 d67 d68
PLACE FOR USE NOW: d90, TRANSITION FOR USE NOW: 42
I am in TRANSITION 42 now, which does TRIGGERING condition.
I mark the OUTPUT place d69, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d69:
   Object id: c5, Object attr value: 165.000000, Domain: fuzzywinddr,
   Fuzzytermlist: southeast south, Membership1: 0.67, Membership2: 1.00

CURRENT_PLACES FOR USE: d49 d50 d51 d52 d53 d54 d55 d56 d57 d60 d61 d62 d63 d64 d65
d66 d67 d68 d69
PLACE FOR USE NOW: d49, TRANSITION FOR USE NOW: 51
I am in TRANSITION 51 now, which does COMPOSITION.
Since composition result is 0, rule r32 is removed from the inference.

CURRENT_PLACES FOR USE: d50 d51 d52 d53 d54 d55 d56 d57 d60 d61 d62 d63 d64 d65 d66
d67 d68 d69
```

```
PLACE FOR USE NOW: d50, TRANSITION FOR USE NOW: 51
Since transition 51 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d51 d52 d53 d54 d55 d56 d57 d60 d61 d62 d63 d64 d65 d66 d67
d68 d69
PLACE FOR USE NOW: d51, TRANSITION FOR USE NOW: 51
Since transition 51 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d52 d53 d54 d55 d56 d57 d60 d61 d62 d63 d64 d65 d66 d67
d68 d69
PLACE FOR USE NOW: d52, TRANSITION FOR USE NOW: 51
Since transition 51 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d53 d54 d55 d56 d57 d60 d61 d62 d63 d64 d65 d66 d67
d68 d69
PLACE FOR USE NOW: d53, TRANSITION FOR USE NOW: 51
Since transition 51 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d54 d55 d56 d57 d60 d61 d62 d63 d64 d65 d66 d67
d68 d69
PLACE FOR USE NOW: d54, TRANSITION FOR USE NOW: 51
Since transition 51 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d55 d56 d57 d60 d61 d62 d63 d64 d65 d66 d67 d68 d69
PLACE FOR USE NOW: d55, TRANSITION FOR USE NOW: 51
Since transition 51 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d56 d57 d60 d61 d62 d63 d64 d65 d66 d67 d68 d69
PLACE FOR USE NOW: d56, TRANSITION FOR USE NOW: 51
Since transition 51 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d57 d60 d61 d62 d63 d64 d65 d66 d67 d68 d69
PLACE FOR USE NOW: d57, TRANSITION FOR USE NOW: 51
Since transition 51 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d60 d61 d62 d63 d64 d65 d66 d67 d68 d69
PLACE FOR USE NOW: d60, TRANSITION FOR USE NOW: 52
```

```
I am in TRANSITION 52 now, which does COMPOSITION.
I mark the OUTPUT place d59, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d59:
   Object id: c5, Membership1: 0.50, Membership2: 1.00

CURRENT_PLACES FOR USE: d61 d62 d63 d64 d65 d66 d67 d68 d69 d59
PLACE FOR USE NOW: d61, TRANSITION FOR USE NOW: 52
Since transition 52 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d62 d63 d64 d65 d66 d67 d68 d69 d59
PLACE FOR USE NOW: d62, TRANSITION FOR USE NOW: 52
Since transition 52 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d63 d64 d65 d66 d67 d68 d69 d59
PLACE FOR USE NOW: d63, TRANSITION FOR USE NOW: 52
Since transition 52 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d64 d65 d66 d67 d68 d69 d59
PLACE FOR USE NOW: d64, TRANSITION FOR USE NOW: 52
Since transition 52 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d65 d66 d67 d68 d69 d59
PLACE FOR USE NOW: d65, TRANSITION FOR USE NOW: 52
Since transition 52 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d66 d67 d68 d69 d59
PLACE FOR USE NOW: d66, TRANSITION FOR USE NOW: 52
Since transition 52 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d67 d68 d69 d59
PLACE FOR USE NOW: d67, TRANSITION FOR USE NOW: 52
Since transition 52 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d68 d69 d59
PLACE FOR USE NOW: d68, TRANSITION FOR USE NOW: 52
Since transition 52 is consumed before, do not fire it again.
```

```
CURRENT_PLACES FOR USE: d69 d59

PLACE FOR USE NOW: d69, TRANSITION FOR USE NOW: 52

Since transition 52 is consumed before, do not fire it again.


CURRENT_PLACES FOR USE: d59

PLACE FOR USE NOW: d59, TRANSITION FOR USE NOW: 85

I am in TRANSITION 85 now, which does CLIPPING.

I mark the OUTPUT place d70, which is added to the tail of

CURRENT places list

TOKEN VALUE put in OUTPUT place d70:

   Object id: c5, Membership1: 0.50


CURRENT_PLACES FOR USE: d70

PLACE FOR USE NOW: d70, TRANSITION FOR USE NOW: 101

I am in TRANSITION 101 now, which does COMBINATION.

I mark the OUTPUT place d106, which is added to the tail of

CURRENT places list

CLIPPED TOKEN VALUE put in OUTPUT place d106:

   Object id: c5, Fuzzytermlist: cloudedstable, Clipping value: 0.50


CURRENT_PLACES FOR USE: d106

PLACE FOR USE NOW: d106, TRANSITION FOR USE NOW: 102

I am in TRANSITION 102 now, which finds the overall FUZZY ACTION.

I mark the OUTPUT place d107, which is added to the tail of

CURRENT places list

TOKEN VALUE put in OUTPUT place d107:

   Object id: c5, Fuzzytermlist: cloudedstable, Overall Clipping value: 0.50
```

B.3 Forecasting the Weather Event - Active Rule Example (2)

```
 INFERENCE GROUP 1 FIRED FOR EXECUTION


CURRENT_PLACES FOR USE: p92 p97 p98 p99
PLACE FOR USE NOW: p92, TRANSITION FOR USE NOW: 1
I am in TRANSITION 1 now, which does FUZZIFICATION.
I mark the OUTPUT place p27, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p27:
   Object id: c5, Object attr value: 4.000000, Domain: fuzzyccolor,
   Fuzzytermlist: grey, Membership: 0.50

CURRENT_PLACES FOR USE: p97 p98 p99 p27
PLACE FOR USE NOW: p97, TRANSITION FOR USE NOW: 7
I am in TRANSITION 7 now, which does FUZZIFICATION.
I mark the OUTPUT place p24, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p24:
   Object id: c5, Object attr value: cloudedstable, Domain: fuzzyweather,
   Fuzzytermlist: cloudedstable, Membership: 0.25

CURRENT_PLACES FOR USE: p98 p99 p27 p24
PLACE FOR USE NOW: p98, TRANSITION FOR USE NOW: 8
I am in TRANSITION 8 now, which does FUZZIFICATION.
Since TRANSITION 8 DOES NOT WORK for INFERENCE GROUP 1, I am not using it.
CURRENT_PLACES FOR USE: p98 p99 p27 p24
PLACE FOR USE NOW: p98, TRANSITION FOR USE NOW: 9
I am in TRANSITION 9 now, which does FUZZIFICATION.
I mark the OUTPUT place p26, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p26:
   Object id: c5, Object attr value: 6.000000, Domain: fuzzywind,
   Fuzzytermlist: mediumstrong, Membership: 0.00

CURRENT_PLACES FOR USE: p99 p27 p24 p26
```

```
PLACE FOR USE NOW: p99, TRANSITION FOR USE NOW: 10
I am in TRANSITION 10 now, which does FUZZIFICATION.
I mark the OUTPUT place p25, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p25:
   Object id: c5, Object attr value: 22.500000, Domain: fuzzywinddr,
   Fuzzytermlist: south southwest, Membership: 0.00

CURRENT_PLACES FOR USE: p99 p27 p24 p26 p25
PLACE FOR USE NOW: p99, TRANSITION FOR USE NOW: 11
I am in TRANSITION 11 now, which does FUZZIFICATION.
I mark the OUTPUT place p38, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p38:
   Object id: c5, Object attr value: 22.500000, Domain: fuzzywinddr,
   Fuzzytermlist: north, Membership: 0.50

CURRENT_PLACES FOR USE: p27 p24 p26 p25 p38
PLACE FOR USE NOW: p27, TRANSITION FOR USE NOW: 45
I am in TRANSITION 45 now, which does COMPOSITION.
Since composition result is 0, rule r5 is removed from the inference.

CURRENT_PLACES FOR USE: p27 p24 p26 p25 p38
PLACE FOR USE NOW: p27, TRANSITION FOR USE NOW: 46
I am in TRANSITION 46 now, which does COMPOSITION.
I mark the OUTPUT place p37, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p37:
   Object id: c5, Membership1: 0.25, Membership2: 0.00

CURRENT_PLACES FOR USE: p24 p26 p25 p38 p37
PLACE FOR USE NOW: p24, TRANSITION FOR USE NOW: 45
Since transition 45 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: p24 p26 p25 p38 p37
PLACE FOR USE NOW: p24, TRANSITION FOR USE NOW: 46
Since transition 46 is consumed before, do not fire it again.
```

```
CURRENT_PLACES FOR USE: p24 p26 p25 p38 p37
PLACE FOR USE NOW: p24, TRANSITION FOR USE NOW: 81
I am in TRANSITION 81 now, which does STRENGTH calculation.
I mark the OUTPUT place p112, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p112:
   Object id: c5, Membership1: 0.25.

CURRENT_PLACES FOR USE: p26 p25 p38 p37 p112
PLACE FOR USE NOW: p26, TRANSITION FOR USE NOW: 45
Since transition 45 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: p25 p38 p37 p112
PLACE FOR USE NOW: p25, TRANSITION FOR USE NOW: 45
Since transition 45 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: p38 p37 p112
PLACE FOR USE NOW: p38, TRANSITION FOR USE NOW: 46
Since transition 46 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: p37 p112
PLACE FOR USE NOW: p37, TRANSITION FOR USE NOW: 76
I am in TRANSITION 76 now, which does STRENGTH calculation.
I mark the OUTPUT place p111, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p111:
   Object id: c5, Membership1: 0.25.

CURRENT_PLACES FOR USE: p112 p111
PLACE FOR USE NOW: p112, TRANSITION FOR USE NOW: 82
I am in TRANSITION 82 now, which does TRIGGERING condition.
I mark the OUTPUT place p46, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p46:
   Object id: c5, Object attr value: 6.000000, Domain: fuzzywind,
   Fuzzytermlist: calm breeze, Membership1: 1.00, Membership2: 0.25
```

```
CURRENT_PLACES FOR USE: p111 p46
PLACE FOR USE NOW: p111, TRANSITION FOR USE NOW: 77
I am in TRANSITION 77 now, which does TRIGGERING condition.
I mark the OUTPUT place p41, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p41:
   Object id: c5, Object attr value: -1.000000, Domain: fuzzytemp,
   Fuzzytermlist: cold, Membership1: 1.00, Membership2: 0.25

CURRENT_PLACES FOR USE: p111 p46 p41
PLACE FOR USE NOW: p111, TRANSITION FOR USE NOW: 78
I am in TRANSITION 78 now, which does TRIGGERING condition.
I mark the OUTPUT place p42, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p42:
   Object id: c5, Object attr value: 165.000000, Domain: fuzzywinddr,
   Fuzzytermlist: south, Membership1: 0.67, Membership2: 0.25

CURRENT_PLACES FOR USE: p111 p46 p41 p42
PLACE FOR USE NOW: p111, TRANSITION FOR USE NOW: 79
I am in TRANSITION 79 now, which does TRIGGERING condition.
I mark the OUTPUT place p43, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p43:
   Object id: c5, Object attr value: 3.000000, Domain: fuzzyccolor,
   Fuzzytermlist: white, Membership1: 0.50, Membership2: 0.25

CURRENT_PLACES FOR USE: p46 p41 p42 p43
PLACE FOR USE NOW: p46, TRANSITION FOR USE NOW: 83
I am in TRANSITION 83 now, which does CLIPPING.
I mark the OUTPUT place p47, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p47:
   Object id: c5, Membership1: 0.25

CURRENT_PLACES FOR USE: p41 p42 p43 p47
```

```
PLACE FOR USE NOW: p41, TRANSITION FOR USE NOW: 50
I am in TRANSITION 50 now, which does COMPOSITION.
I mark the OUTPUT place p40, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p40:
   Object id: c5, Membership1: 0.50, Membership2: 0.25

CURRENT_PLACES FOR USE: p42 p43 p47 p40
PLACE FOR USE NOW: p42, TRANSITION FOR USE NOW: 50
Since transition 50 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: p43 p47 p40
PLACE FOR USE NOW: p43, TRANSITION FOR USE NOW: 50
Since transition 50 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: p47 p40
PLACE FOR USE NOW: p47, TRANSITION FOR USE NOW: 94
I am in TRANSITION 94 now, which does COMBINATION.
I mark the OUTPUT place p102, which is added to the tail of
CURRENT places list
CLIPPED TOKEN VALUE put in OUTPUT place p102:
   Object id: c5, Fuzzytermlist: fog, Clipping value: 0.25

CURRENT_PLACES FOR USE: p40 p102
PLACE FOR USE NOW: p40, TRANSITION FOR USE NOW: 80
I am in TRANSITION 80 now, which does CLIPPING.
I mark the OUTPUT place p44, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p44:
   Object id: c5, Membership1: 0.12

CURRENT_PLACES FOR USE: p102 p44
PLACE FOR USE NOW: p102, TRANSITION FOR USE NOW: 95
I am in TRANSITION 95 now, which finds the overall FUZZY ACTION.
I am waiting for the other clipped actions to be combined

CURRENT_PLACES FOR USE: p44 p102
```

```
PLACE FOR USE NOW: p44, TRANSITION FOR USE NOW: 93
I am in TRANSITION 93 now, which does COMBINATION.
I mark the OUTPUT place p102, which is added to the tail of
CURRENT places list
CLIPPED TOKEN VALUE put in OUTPUT place p102:
   Object id: c5, Fuzzytermlist: snow, Clipping value: 0.12

CURRENT_PLACES FOR USE: p102
PLACE FOR USE NOW: p102, TRANSITION FOR USE NOW: 95
I am in TRANSITION 95 now, which finds the overall FUZZY ACTION.
I mark the OUTPUT place p103, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place p103:
   Object id: c5, Fuzzytermlist: fog, Overall Clipping value: 0.25
```

B.4 Querying the Temperature Change Forecast - Deductive Rule Example (2)

```
INFERENCE GROUP 2 FIRED FOR EXECUTION

CURRENT_PLACES FOR USE: d89
PLACE FOR USE NOW: d89, TRANSITION FOR USE NOW: 12
I am in TRANSITION 12 now, which does TRIGGERING condition.
I mark the OUTPUT place d72, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d72:
   Object id: c5, Object attr value: cloudedstable, Domain: fuzzyweather,
   Fuzzytermlist: clear clearfew cloudedinstable, Membership1: 0.50, Membership2: 1.00

CURRENT_PLACES FOR USE: d89 d72
PLACE FOR USE NOW: d89, TRANSITION FOR USE NOW: 13
I am in TRANSITION 13 now, which does TRIGGERING condition.
I mark the OUTPUT place d73, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d73:
   Object id: c5, Object attr value: 3.500000, Domain: fuzzypreschdir,
   Fuzzytermlist: decreasing, Membership1: 0.00, Membership2: 1.00

CURRENT_PLACES FOR USE: d89 d72 d73
PLACE FOR USE NOW: d89, TRANSITION FOR USE NOW: 14
I am in TRANSITION 14 now, which does TRIGGERING condition.
I mark the OUTPUT place d74, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d74:
   Object id: c5, Object attr value: 22.500000, Domain: fuzzywinddr,
   Fuzzytermlist: south southwest, Membership1: 0.00, Membership2: 1.00

CURRENT_PLACES FOR USE: d89 d72 d73 d74
PLACE FOR USE NOW: d89, TRANSITION FOR USE NOW: 15
I am in TRANSITION 15 now, which does TRIGGERING condition.
I mark the OUTPUT place d75, which is added to the tail of
CURRENT places list
```

```
TOKEN VALUE put in OUTPUT place d75:
   Object id: c5, Object attr value: 165.000000, Domain: fuzzywinddr,
   Fuzzytermlist: north northwest northeast, Membership1: 0.00, Membership2: 1.00

CURRENT_PLACES FOR USE: d89 d72 d73 d74 d75
PLACE FOR USE NOW: d89, TRANSITION FOR USE NOW: 16
I am in TRANSITION 16 now, which does TRIGGERING condition.
I mark the OUTPUT place d78, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d78:
   Object id: c5, Object attr value: cloudedstable, Domain: fuzzyweather,
   Fuzzytermlist: cloudedstable, Membership1: 1.00, Membership2: 1.00

CURRENT_PLACES FOR USE: d89 d72 d73 d74 d75 d78
PLACE FOR USE NOW: d89, TRANSITION FOR USE NOW: 17
I am in TRANSITION 17 now, which does TRIGGERING condition.
I mark the OUTPUT place d79, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d79:
   Object id: c5, Object attr value: 3.500000, Domain: fuzzypreschdir,
   Fuzzytermlist: increasing, Membership1: 1.00, Membership2: 1.00

CURRENT_PLACES FOR USE: d89 d72 d73 d74 d75 d78 d79
PLACE FOR USE NOW: d89, TRANSITION FOR USE NOW: 18
I am in TRANSITION 18 now, which does TRIGGERING condition.
I mark the OUTPUT place d80, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d80:
   Object id: c5, Object attr value: 22.500000, Domain: fuzzywinddr,
   Fuzzytermlist: north northwest northeast, Membership1: 0.50, Membership2: 1.00

CURRENT_PLACES FOR USE: d89 d72 d73 d74 d75 d78 d79 d80
PLACE FOR USE NOW: d89, TRANSITION FOR USE NOW: 19
I am in TRANSITION 19 now, which does TRIGGERING condition.
I mark the OUTPUT place d81, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d81:
   Object id: c5, Object attr value: 165.000000, Domain: fuzzywinddr,

   Fuzzytermlist: south southwest, Membership1: 0.67, Membership2: 1.00

CURRENT_PLACES FOR USE: d89 d72 d73 d74 d75 d78 d79 d80 d81
PLACE FOR USE NOW: d89, TRANSITION FOR USE NOW: 20
I am in TRANSITION 20 now, which does TRIGGERING condition.
I mark the OUTPUT place d84, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d84:
   Object id: c5, Object attr value: cloudedstable, Domain: fuzzyweather,
   Fuzzytermlist: clear clearfew cloudedinstable cloudedstable, Membership1: 1.00,
   Membership2: 1.00

CURRENT_PLACES FOR USE: d89 d72 d73 d74 d75 d78 d79 d80 d81 d84
PLACE FOR USE NOW: d89, TRANSITION FOR USE NOW: 21
I am in TRANSITION 21 now, which does TRIGGERING condition.
I mark the OUTPUT place d85, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d85:
   Object id: c5, Object attr value: 3.500000, Domain: fuzzypreschdir,
   Fuzzytermlist: nochange, Membership1: 0.00, Membership2: 1.00

CURRENT_PLACES FOR USE: d89 d72 d73 d74 d75 d78 d79 d80 d81 d84 d85
PLACE FOR USE NOW: d89, TRANSITION FOR USE NOW: 22
I am in TRANSITION 22 now, which does TRIGGERING condition.
I mark the OUTPUT place d86, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d86:
   Object id: c5, Object attr value: 22.500000, Domain: fuzzywinddr, F
   uzzytermlist: west, Membership1: 0.00, Membership2: 1.00

CURRENT_PLACES FOR USE: d89 d72 d73 d74 d75 d78 d79 d80 d81 d84 d85 d86
PLACE FOR USE NOW: d89, TRANSITION FOR USE NOW: 23
I am in TRANSITION 23 now, which does TRIGGERING condition.
I mark the OUTPUT place d87, which is added to the tail of
CURRENT places list
```

```
TOKEN VALUE put in OUTPUT place d87:
   Object id: c5, Object attr value: 6.000000, Domain: fuzzywind,
   Fuzzytermlist: breeze, Membership1: 1.00, Membership2: 1.00

CURRENT_PLACES FOR USE: d72 d73 d74 d75 d78 d79 d80 d81 d84 d85 d86 d87
PLACE FOR USE NOW: d72, TRANSITION FOR USE NOW: 53
I am in TRANSITION 53 now, which does COMPOSITION.
Since composition result is 0, rule r102 is removed from the inference.

CURRENT_PLACES FOR USE: d73 d74 d75 d78 d79 d80 d81 d84 d85 d86 d87
PLACE FOR USE NOW: d73, TRANSITION FOR USE NOW: 53
Since transition 53 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d74 d75 d78 d79 d80 d81 d84 d85 d86 d87
PLACE FOR USE NOW: d74, TRANSITION FOR USE NOW: 53
Since transition 53 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d75 d78 d79 d80 d81 d84 d85 d86 d87
PLACE FOR USE NOW: d75, TRANSITION FOR USE NOW: 53
Since transition 53 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d78 d79 d80 d81 d84 d85 d86 d87
PLACE FOR USE NOW: d78, TRANSITION FOR USE NOW: 54
I am in TRANSITION 54 now, which does COMPOSITION.
I mark the OUTPUT place d77, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d77:
   Object id: c5, Membership1: 0.50, Membership2: 1.00

CURRENT_PLACES FOR USE: d79 d80 d81 d84 d85 d86 d87 d77
PLACE FOR USE NOW: d79, TRANSITION FOR USE NOW: 54
Since transition 54 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d80 d81 d84 d85 d86 d87 d77
PLACE FOR USE NOW: d80, TRANSITION FOR USE NOW: 54
Since transition 54 is consumed before, do not fire it again.
```

```
CURRENT_PLACES FOR USE: d81 d84 d85 d86 d87 d77

PLACE FOR USE NOW: d81, TRANSITION FOR USE NOW: 54

Since transition 54 is consumed before, do not fire it again.


CURRENT_PLACES FOR USE: d84 d85 d86 d87 d77

PLACE FOR USE NOW: d84, TRANSITION FOR USE NOW: 55

I am in TRANSITION 55 now, which does COMPOSITION.

Since composition result is 0, rule r122 is removed from the inference.


CURRENT_PLACES FOR USE: d85 d86 d87 d77

PLACE FOR USE NOW: d85, TRANSITION FOR USE NOW: 55

Since transition 55 is consumed before, do not fire it again.


CURRENT_PLACES FOR USE: d86 d87 d77

PLACE FOR USE NOW: d86, TRANSITION FOR USE NOW: 55

Since transition 55 is consumed before, do not fire it again.


CURRENT_PLACES FOR USE: d87 d77

PLACE FOR USE NOW: d87, TRANSITION FOR USE NOW: 55

Since transition 55 is consumed before, do not fire it again.


CURRENT_PLACES FOR USE: d77

PLACE FOR USE NOW: d77, TRANSITION FOR USE NOW: 87

I am in TRANSITION 87 now, which does CLIPPING.

I mark the OUTPUT place d82, which is added to the tail of

CURRENT places list

TOKEN VALUE put in OUTPUT place d82:

   Object id: c5, Membership1: 0.50


CURRENT_PLACES FOR USE: d82

PLACE FOR USE NOW: d82, TRANSITION FOR USE NOW: 97

I am in TRANSITION 97 now, which does COMBINATION.

I mark the OUTPUT place d104, which is added to the tail of

CURRENT places list

CLIPPED TOKEN VALUE put in OUTPUT place d104:

   Object id: c5, Fuzzytermlist: decrease, Clipping value: 0.50
```

```
CURRENT_PLACES FOR USE: d104

PLACE FOR USE NOW: d104, TRANSITION FOR USE NOW: 99

I am in TRANSITION 99 now, which finds the overall FUZZY ACTION.

I mark the OUTPUT place d105, which is added to the tail of

CURRENT places list

TOKEN VALUE put in OUTPUT place d105:

   Object id: c5, Fuzzytermlist: decrease, Overall Clipping value: 0.50
```

APPENDIX C

ITASCA OBJECT DATABASE SYSTEM

C.1 Overview

ITASCA employs distributed architecture with private and shared objects spread across multiple nodes on a local-area network. The ITASCA model follows the object-oriented view thjat uniformly models any real-world entity as an object. Each object has a unique identifier along with a state and behavior. Attributes represent the state of an object while methods define th behavior. A class collects objects that share the same set of attributes and methods. Subclasses derive ffrom existing classes. The resulting schema, or database definition, is the class hierarchy. Each subclass inherits all the attributes and methods of its superclasses. ITASCA supports multiple inheritance, so a subclass may derive from more than one class.

Dynamic Schema Modification: ITASCA has a rich capability to change the schema or datbase definition dynamically. This provides flexibility for application development, maintenance, and modification. ITASCA class definitions, inheritance structure, attribute specifications and methods can be modified without requiring application shutdown, databse shutdown or off-loading of data. Dynamic schema modification can significantly reduce costs. It is also useful in environments where changes to data definitions are normal or relatively frequent.

ITASCA stores and executes method code in the database. Methods are managed as a part of the ITAASCA schema. The code executes within the database opposed to in the application. This allows:

- Modification of method code without recompilation of application code.
- Reuse of objects among multiple programming languaages.
- Reuse of server methods among programming languages.

Language Independence: C++, C, CLOS, Smalltalk and Lisp applications can access and update the objects and invoke the same methods stored in ITASCA. The capability of storing methods in the database eliminates any restriction on programming languages. This provides increased flexibility for development, maintenance and modification.

Extensible Architecture: ITASCA allows the refinement of kernel methods at the class level. This allows customizing the behavior for making instances, deleting instances, checking objectsin and out of the shared database, making new instance versions. This extensibility simplifies modification and maintenance of systems. Existing applications do not need to be modified when new behavior is added to a kernel method.

Object Data Model: ITASCA supports accepted object-oriented principles, including data abstraction, encapsulation, inheritance, object identification and classes. Itasca incorporates these principles into the persistent and shared environment of a distributed database manager.

Data Abstraction: ITASCA supports data abstraction, i.e. the capabilirt to define data structures composed of a variety of data types. It allows the use of abstract data structures such as binary trees, stacks, queues, graphs and so on.

Encapsulation: Encapsulation of data and operations is similar to the software engineering concept of information hiding. This concept states that modules should contain or hide information (code and data). It also states that all operations on the data must use an external interface. Modules in an object databse are objects that contain both methods and attributes. Operations on objects are performed by sending messages to the objects. An object reacts to the message by executing the corresponding method or by retaining the appropriate attribute value. A method can be as simple as editing a data value or it can be a complex algorithm. A method in ITASCA can also call existing C or Lisp code directly. An ITASCA object encapsulates all the data as well as the methods that affect the data. ITASCA stores both the data and the methods in the database.

Inheritance: ITASCA users derive new classes from existing classes. The new class is a subclass and inherits all the attributes and methods of the existing class. In ITASCA, a class can have more than one superclass (multiple inheritance). Inheritance provides explicit support for reusable code and data. Refining inherited methods adds new functionality to the existing code or methods.

Object Identifier: ITASCA assigns each object a system-supplied unique object identifier. ITASCA supports uniqueness of data but such data does not need to have a self-contained unique identifier. This allows the construction of an abstract data structure without requiring the application to manufucture unnecessary unique keys. ITASCA automatically supplies a unique identifier for absrract structures and stores them in the database. The ITASCA OID encodes the private database where each class and instance is created.

Class: All objects that share the same set of attributes and methods are grouped in a class. An instance object may belong to only one class. The relation between an instance object and its class is the instance-of-relationship. A class is anologues to an abstract data type. A class may also be primitive, meaning that has associated instances but does not have associated attributes.

Persistence: ITASCA stores complex data structures directly without conversion to a different storage representation. It is transparent to the application/user whether a referenced object is in memory or on disk. An object persists by default. Existence of an object can be dependent upon the existence of one or other objects.

Multiple Servers: A server can service multiple clients with one server process per machine. Servers maintain a page buffer and an object buffer; objects in the page buffer move to the object buffer. The server maintains an object buffer for concurrency control and locking.

Multiple Clients: A single client can access multiple servers with one client process per machine and multiple concurrent sessions per client process. Clients maintain an object buffer for objects in local memory. It is possible to have thousands of frequently used objects cached in the local object buffer.

No single point of failure: ITASCA does not rely on central services for class naming, resource allocation, deadlock detection, or schema information. The loss of any machine on the network will not cause to entire system to become unavailable. Only the failed machine's data becomes unavailable. Restoring the machine to service makes the data available. Work in both the shared and private databases associated with other machines can continue if the unavailablae data has no impact on the work. Even changes to replicated objects (such as the schema) can occur because the ITASCA spools such chanages for later use by the recovering site.

Communication Subsystem: ITASCA uses the TCP/IP protocol over UNIX sockets. It supports multicast and broadcast through remote procedure calls with immediate return (i.e. without blocking). Multiple applications can participate in the same transaction through the use of shared sessions.

Shared Database: ITASCA handles all aspects of managing objects in the distributed database. One partition of the distributed databse exists on each server site. The user or application does not need to know the location af a targeted object in the database: the in-memory movement of objects is transparent. ITASCA also supports explicitly moving objects from one site to another.

Dynamic Configuration: A user can create, migrate, or destroy distributed sites at any time given proper authorization. There is no need to halt any processing to reconfigure the distributed database system.

Private Databases: Multiple private databases can exist at a server site and data can be moved between the shared database and those private databases. Enhanced locality of reference in a private database can convert a possible distributed transaction to a local transaction. This reduces the need for authorization, concurrency control and locking. Authorization is at tahe private databse level for objects in the private database and not at the object level. A work group may also share a private database. Private databases are part of long-duration transactions.

Transaction Management: Long-duration transactions start with a checkout followed by multiple commits or aborts in the private database and end with check-in. The methods for check-out and check-in can be refined to perform additional application specific actions.

Short Transactions: A short transaction is a sequence of operations grouped into an atomic or indivisible operation. Short transactions are atomic, consistent, independent and durable. If a transaction aborts, or fails to complete, none of the changes appear in the persistent database. When multiple transactions are executing concurrently, the databse maintains a consistent state. This prevents changes by one transaction from interacting in an uncontrolled manner with changes by another transaction. If multiple users are sharing the database, each users transaction is independent of the other users' transactions. Process or storage media falures do not affect completed transactions. Completed transactions are durable.

Concurrency Control: Pessimistic concurrency control implements serializability, allowing simultaneous, independent transactions to execute in parallel. This allows transactions to request a lock for concurrent access on the same objects. A transaction may wait for a lock to become available until ITASCA detects a deadlock. The length of the deadlock time out can be specified by the user. When ITASCA detects a deadlock, control is passed back to the user/application indicating the last action was cancelled. The user application can decide to reexecute or abort the transaction.

Two-phase Commit: ITASCA uses a two-phase commit protocol to ensure that every distributed databse transaction is atomic and durabale.

Long-duration Transactions: Checking out an object from the shared database to a private database starts a long duration transaction for that object. Checking the object into the shared database ends the long duration transaction for the object. Long duration transactions can last any length of time. Any number of short duration transactions can occur between any given check-out and check-in.

Any number of checkouts can occur before a check-in. Any number of checkins can occur after a single checkout. Check-in/check-out can be done with a general query as well as by identifying specific objects. Composite objects' checkin check-out is in the same way as simple single objects. Non-versioned objects physically move between shared and private databases during checkout/checkin. Versioned objects have new versions made in the target private database at checkout time leaving the parent object in the shared database. Refinement of check-in/checkout methods allow additional class specific operations.

Concurrency Control Goals: There are several concurrency control goals in ITASCA's architecture. First, object distribution is invisible to the user as much as possible. Second, the system strives for maximal autonomy. Third, ITASCA shields each private database from all other private databases. Fourth, transactions involving only objects in a private database may be committed locally.

Sessions: A session encapsulates a sequence of transactions. Multiple independent sessions may exist on the same workstation in ITASCA. Multiple applications may share the same session. Shared sessions in ITASCA allow multiple processes to share a transaction. Multiple applications participating in the same session share the same locks.

Cooperative Transactions: ITASCA allows multiple users to participate in the same transaction for cooperative work. The users have the option of being in the same or seperate sessions. Seperate sessions enforce locking.

Locking: ITASCA uses two-phase locking for the shared database at the object instance level of granularity. Locks change dynamically. Composite objects are not a single logical entity for locking. The components of a composite object may be in several partitions of the shared databse on different sites. ITASCA shields private databases from each other and from the shared database.

CPU Failure Recovery: A reliable form of logging for a distributed databse environment, ITASCA uses undo logging for implementation simplicity, log space and update pattern of ITASCA applications. CPU failure recovery uses undo logging, which keeps the inverse of changes in an incremental log. ITASCA applies this log to the database at recovery time. Recovery goes backwards from the state at the time of the faailure to a consistent databse state. The undo operation will roll back incomplete transactions to get the end of the last completed transaction. When a transaction commits, the system flushes all updated objects from the object buffer to the page buffer. Log and data pages are then forced to disk.

Media Failure Recovery: Optional mirror wrting of data provides for protection against media failure. Upon a primary or backup disk failure, ITASCA issues a warning and sends electronic mail to the system manager warning of the failure. Since a method executes this action, changing the method will allow additional actions at the warning of a disk failure.

Single Schema: ITASCA uses one schema for the entire distributed database, and each private database has a corresponding subschema. Each site in the distributed database has a copy of the shared database schema, including the code for methods. At execution time only data moves among sites. This dramatically improves application performance.

VITA

Burçin Bostan was born in Akşehir on June 15, 1972. She received her B.S. degree in Computer Engineering from Middle East Technical University in July 1994. She received her M.S. degree from the same department in January 1997. She worked at the Computer Engineering Department of Middle East Technical University as a teaching assistant between 1994-1997. Since 1997 she has been working at Payment Systems Department of the Central Bank of Turkey. She has participated in analysis, design and development of the 2^{nd} generation Electronic Funds Transfer System (EFT) and the Electronic Securities Transfer System of Turkey at Logica CMG UK, London between April 1998 and June 1999. Her research interests include fuzzy logic, object-oriented databases and knowledge base systems.

Publications

1. Bostan-Korpeoglu B. and Yazici A., "Incorporating Fuzziness into Active Rules", submitted for publication in *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems*, 24 October 2004.

2. Bostan-Korpeoglu B. and Yazici A., "A Fuzzy Petri Net Model for Intelligent Databases", submitted for publication in *Data & Knowledge Engineering Journal*, 16 October 2004.

3. Bostan-Korpeoglu B. and Yazici A., "Using Fuzzy Petri Nets for Static Analysis of Rule-Bases", in *Proceedings of 19th International Symposium on*

Computer and Information Sciences, (ISCIS'04), (Springer Verlag LNCS 3280), Antalya, Turkey, 27-29 October, 2004.

4. Bostan-Korpeoglu B. and Yazici A., "An Active Object-Oriented Database Approach", in *Proceedings of 13th IEEE International Conference on Fuzzy Systems, (FUZZ-IEEE'04)*, Budapest, Hungary, 25-29 July, 2004.

5. Bostan B. and Yazici A., "Fuzzy Inference Mechanism in Fuzzy ECA Rules", in *Proceedings of 10th International Fuzzy System Association World Congress, (IFSA'03)*, Istanbul, Turkey, June 29-July 2, 2003.

6. Bostan B. and Yazici A., "Fuzzy Deductive Object-Oriented Database Model", in *Journal of Elektrik* published by TÜBİTAK (Research Institute of Science and Technology of Turkey), July 1998.

7. Bostan B. and Yazici A., "A Fuzzy Deductive Object-Oriented Data Model", in *Proceedings of 7th IEEE International Conference on Fuzzy Systems, (FUZZ-IEEE'98)*, Vol.2, pp. 1361-1366, Alasca, USA, May 1998.

8. Bostan B. and Yazici A., "UDOOM: Uncertainty in Deductive Object-Oriented Data Model", in *Proceedings of 11th International Symposium on Computer and Information Sciences, (ISCIS'96)*, pp.113-122, Antalya, Turkey, November 1996.