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 Yazar
Supervisor

Exchanging Committee Members

Prof. Dr. Ö zgür Ulusoy (Bilkent University)

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

Name, Lastname: Burçin Bostan

Signature: 

iii
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 occurrences 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 PETRI NET
MODELİ

Bostan, Burçin
Doktora, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Adnan Yazarı

Mart 2005, 181 sayfa


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-yöneltik veritabanı, bilgi-tabanı, bulanıklık, aktif veritabanı, bulanık Petri net, statik kural analizi, sonlanma, tutarlılık
To my parents, Berrin - Ahmet Bostan
and
my lovely grandmother, Aife 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. Ö zgür Ulusov, 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 İ lker 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

PLAGIARISM ......................................................... iii
ABSTRACT ............................................................... iv
Dedication .............................................................. vi
ACKNOWLEDGMENTS .................................................... ix
TABLE OF CONTENTS .................................................. x
LIST OF TABLES ....................................................... xiv
LIST OF FIGURES ...................................................... xv
LIST OF ABBREVIATIONS .............................................. xvii

## CHAPTER

1 INTRODUCTION .................................................... 1
   1.1 Scope of the Study ............................................ 1
   1.2 Power of Petri Net within Alternatives ....................... 6
   1.3 Summary of Contributions .................................... 7
   1.4 Organization of the Thesis ................................... 8
2 BACKGROUND AND RELATED WORK ................................. 9
   2.1 Basic Concepts of Fuzzy Logic ................................. 9
       2.1.1 Fuzzy Logic in Database Systems ....................... 13
           2.1.1.1 Approaches for Representing Imprecise Attribute Values ...................... 13
           2.1.1.2 Partial Membership in a Relation ................................. 14
   2.2 Object-oriented Database Model ............................... 15
       2.2.1 Fuzzy Logic in Object-oriented Database Model ......... 16
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3</td>
<td>Aspects of Active Databases</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>Petri Nets</td>
<td>19</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Regular Petri Nets</td>
<td>19</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Colored Petri Nets</td>
<td>20</td>
</tr>
<tr>
<td>2.4.3</td>
<td>Fuzzy Petri Nets</td>
<td>21</td>
</tr>
<tr>
<td>2.4.3.1</td>
<td>Fuzzy Petri Nets for Dealing with Fuzzy Deductive Rules</td>
<td>21</td>
</tr>
<tr>
<td>2.5</td>
<td>Static Rule Analysis</td>
<td>22</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Termination Analysis</td>
<td>22</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Confluence Analysis</td>
<td>24</td>
</tr>
<tr>
<td>2.6</td>
<td>Previous Studies and Contribution of This Study</td>
<td>26</td>
</tr>
<tr>
<td>2.6.1</td>
<td>Active Studies</td>
<td>26</td>
</tr>
<tr>
<td>2.6.1.1</td>
<td>Active Object-Oriented Databases</td>
<td>26</td>
</tr>
<tr>
<td>2.6.1.2</td>
<td>Fuzziness in Active Databases</td>
<td>28</td>
</tr>
<tr>
<td>2.6.1.3</td>
<td>Our Study vs Active Studies</td>
<td>28</td>
</tr>
<tr>
<td>2.6.2</td>
<td>Fuzzy Petri Nets and Static Rule Analysis</td>
<td>30</td>
</tr>
<tr>
<td>2.6.2.1</td>
<td>Fuzzy Petri Nets</td>
<td>30</td>
</tr>
<tr>
<td>2.6.2.2</td>
<td>Static Rule Analysis</td>
<td>30</td>
</tr>
<tr>
<td>2.6.2.3</td>
<td>Our Study vs Other Studies on FPNs</td>
<td>31</td>
</tr>
<tr>
<td>3</td>
<td>INCORPORATING FUZZINESS INTO ACTIVE RULES</td>
<td>33</td>
</tr>
<tr>
<td>3.1</td>
<td>Inter-rule Fuzziness via Scenarios</td>
<td>33</td>
</tr>
<tr>
<td>3.2</td>
<td>Incorporating Fuzzy Inference into Active Rules</td>
<td>34</td>
</tr>
<tr>
<td>3.3</td>
<td>A model for the Similarity of the Rule to the Scenario</td>
<td>37</td>
</tr>
<tr>
<td>3.4</td>
<td>Example - An Overheating Alarm Generation in a Drive System of a Couple of Motors</td>
<td>37</td>
</tr>
<tr>
<td>4</td>
<td>FUZZY PETRI NETS FOR MODELING FUZZY RULE-BASED REASONING</td>
<td>44</td>
</tr>
<tr>
<td>4.1</td>
<td>Fuzzy Petri Nets for Fuzzy Rules</td>
<td>44</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Mapping Fuzzy Rules to Fuzzy Petri Net</td>
<td>46</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Constructing the FPN</td>
<td>49</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Example 1 - An Overheating Alarm Generation in a Drive System of 1 Motor (active rule example)</td>
<td>52</td>
</tr>
<tr>
<td>4.1.4</td>
<td>Example 2 - Automatic Selection of Washing Cycle for a Washing Machine (deductive rule example)</td>
<td>54</td>
</tr>
</tbody>
</table>
5 USING FUZZY PETRI NETS FOR STATIC ANALYSIS OF RULE-BASES ........................................ 60

5.1 An Algorithm for Static Analysis on the FPN .......................................................... 60
5.1.1 How to Determine Cycles and True cycles ....................................................... 62
5.1.2 Example ........................................................................................................... 63
5.1.3 The Algorithm for Determining the True Cycles ............................................. 64

5.2 Guaranteeing Confluent Executions in Our FPN Model ......................................... 66

6 IMPLEMENTATION ................................................................................................. 70

6.1 Architecture of the system using FPN based inference mechanism .................. 70

6.2 Implementation Platform ...................................................................................... 71

6.3 Module Structure of the Prototype System ......................................................... 72

6.3.1 Fuzzy Definitions ............................................................................................. 72
6.3.2 Class Operations ............................................................................................... 75
6.3.2.1 Derived Attributes ..................................................................................... 77
6.3.3 Object Operations ............................................................................................ 78
6.3.4 Rules .................................................................................................................. 79
6.3.5 Queries .............................................................................................................. 80
6.3.6 Computational Complexities of the Algorithms ............................................. 86
6.3.6.1 Construction of the FPN .......................................................................... 86
6.3.6.2 Determining true cycles ........................................................................... 90
6.3.6.3 Checking the Confluence ......................................................................... 91
6.3.6.4 Inference Algorithm .................................................................................. 91

7 AN APPLICATION EXAMPLE:
WEATHER FORECASTING ..................................................................................... 94

7.1 What is Included in Our Forecast ........................................................................ 94
7.1.1 Season Scenarios ............................................................................................ 96

7.2 What "Weather Lore" is ....................................................................................... 96
7.3 Why Fuzzy Logic is Suitable for Predicting the Weather ............................... 97
7.4 Active Rules and FPN Construction ................................................................. 98
7.4.1 1st Level Rules - Rules for Forecasting the Weather ................................. 98
7.4.2 2nd Level Rules ............................................................................................... 102
7.4.2.1 Rules for Forecasting the Weather Event 102
7.4.2.2  Rules for Forecasting the Temperature Change .......................... 104

7.5  Deductive Rules and FPN Construction ................................. 107
7.5.1  Rules for querying the weather forecast ......................... 107
7.5.2  Rules for querying the weather event forecast ............... 109
7.5.3  Rules for querying the temperature change forecast 111

7.6  Examples .............................................................. 112
7.6.1  Forecasting the Weather - Active Rule Example(1) 119
7.6.2  Querying the Weather Forecast - Deductive Rule
       Example (1) ..................................................... 121
7.6.3  Forecasting the Weather Event - Active Rule Ex-
       ample (2) ..................................................... 122
7.6.4  Querying the Temperature Change Forecast - De-
      ductive Rule Example (2) .................................... 125

8  CONCLUSION AND FUTURE WORK ........................................... 129

REFERENCES ................................................................. 133

APPENDICES ................................................................. 140

A  CLASS DEFINITIONS AND FUZZY DOMAINS OF METEORO-
    ROLOGICAL DATA .................................................... 140

B  APPLICATION LOG FOR THE EXAMPLES .................................. 146
B.1  Forecasting the Weather - Active Rule Example(1) ............. 146
B.2  Querying the Weather Forecast - Deductive Rule Example
       (1) ............................................................... 153
B.3  Forecasting the Weather Event - Active Rule Example (2) 161
B.4  Querying the Temperature Change Forecast - Deductive
       Rule Example (2) ............................................. 167

C  ITASCA OBJECT DATABASE SYSTEM ...................................... 173
C.1  Overview ............................................................. 173

VITA ................................................................. 180
# LIST OF TABLES

2.1 Similarity Matrix for Age Attribute .............................................. 14

3.1 Similarity Matrix for Temperature ................................................. 40
3.2 Similarity Matrix for AlarmNotification ......................................... 40
3.3 Similarity Matrix for Quantifiers .................................................. 41

7.1 Season Scenarios .............................................................................. 96
7.2 Abbreviations used in the Petri Net Diagram .................................... 114
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 .................................................................................. 119
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 .................................................................................. 125

A.1 Membership Functions for pressure value attribute ......................... 142
A.2 Membership Functions for pressure change velocity attribute ........... 142
A.3 Membership Functions for pressure change direction attribute .......... 142
A.4 Membership Functions for temperature value attribute .................... 142
A.5 Membership Functions for temperature change velocity attribute ...... 142
A.6 Membership Functions for temperature change direction attribute ...... 142
A.7 Membership Functions for humidity value attribute ......................... 143
A.8 Membership Functions for humidity change velocity attribute .......... 143
A.9 Membership Functions for humidity change direction attribute .......... 143
A.10 Membership Functions for wind value attribute ............................... 143
A.11 Membership Functions for wind value change velocity attribute ....... 143
A.12 Membership Functions for wind direction attribute .......................... 144
A.13 Membership Functions for wind direction change velocity attribute ... 144
A.14 Membership Functions for cloud cover attribute ............................. 144
A.15 Membership Functions for cloud base attribute .............................. 144
A.16 Membership Functions for cloud base change direction attribute ...... 144
A.17 Membership Functions for cloud base change velocity attribute ...... 145
A.18 Membership Functions for cloud orientation attribute ..................... 145
LIST OF FIGURES

2.1 Membership Graph of Temperature Attribute ............................. 10
2.2 Possibility distribution of the fuzzy set young .......................... 12
2.3 Phases of rule execution ............................................... 18
2.4 Firing a Regular Petri Net ............................................... 20
2.5 Firing a Fuzzy Petri Net ............................................... 22
2.6 A knowledge base and its triggering and activation graph. ............ 24
2.7 Rule execution by means of rule execution sequences .................. 25
2.8 Commutativity of a rule pair .......................................... 26

4.1 Firing the Fuzzy Petri Net. ................................................. 46
4.2 Modeling Fuzzy Inference using the Fuzzy Petri Net ..................... 50
4.3 Construct_FPN Algorithm ................................................. 51
4.4 Place types used in the FPN .............................................. 52
4.5 Modeling alarm treatment in an industrial drive control system by using FPN ................................................................. 54
4.6 Modeling washing cycle selection in a washing machine by using FPN ................................................................. 57

5.1 Static Analysis Algorithm .................................................. 61
5.2 Dependencies and the Triggering Graph for the Example ............... 63
5.3 FPN constructed for the Example ......................................... 64
5.4 Determine_true_cycles Algorithm ....................................... 65
5.5 Check_Confluence Algorithm ............................................. 67
5.6 Inference Algorithm ....................................................... 68

6.1 Architecture of the System ................................................ 71
6.2 The implementation platform of the prototype system .................... 72
6.3 Module Structure of the prototype system ................................ 73
6.4 Fuzzy type definition ..................................................... 74
6.5 Similarity matrix definition ............................................. 74
6.6 Membership function definition ......................................... 75
6.7 Class creation (1) ....................................................... 76
6.8 Attribute Definition - Class Creation(2) ............................... 76
6.9 Derived Attribute Definition - Class Creation(3) ...................... 78
6.10 Object creation ........................................................ 79
6.11 Object update .......................................................... 80
6.12 Rule operations (1) ..................................................... 81
# LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN</td>
<td>Petri Net</td>
</tr>
<tr>
<td>FPN</td>
<td>Fuzzy Petri Net</td>
</tr>
<tr>
<td>CPN</td>
<td>Colored Petri Net</td>
</tr>
<tr>
<td>TG</td>
<td>Triggering Graph</td>
</tr>
<tr>
<td>AG</td>
<td>Activation Graph</td>
</tr>
</tbody>
</table>
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 challenging 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 extension 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 data-
log 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 auto-
matically 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, in-
ternally 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 con-
sidered 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 sequentiailty,
  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:

- P'Ns' 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 algorithmically 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 sup-
ported. They have to be extended with a data structure that remembers the
event parameters of the primitive events from the time of their occurrence 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.

7
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
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 $\mu_A$ 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 $\mu_A$ is defined as follows:

$$\mu_A(x) : U \longrightarrow [0, 1] \quad (2.1)$$

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:

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

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 $28^\circ C$ 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. $\Pi$ 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

$$
\Pi_X(x) = \mu_A(x)
$$
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 features: 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

<table>
<thead>
<tr>
<th>Age</th>
<th>child</th>
<th>young</th>
<th>middle-aged</th>
<th>old</th>
</tr>
</thead>
<tbody>
<tr>
<td>child</td>
<td>1</td>
<td>0.7</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>young</td>
<td>0.7</td>
<td>1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>middle-aged</td>
<td>0.2</td>
<td>0.2</td>
<td>1</td>
<td>0.9</td>
</tr>
<tr>
<td>old</td>
<td>0.2</td>
<td>0.2</td>
<td>0.9</td>
<td>1</td>
</tr>
</tbody>
</table>

Pomposibility-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:

$$Possibility(young|\Pi_{age}(suspect)) = Max(\min(0.5, 1), \min(1, 0.8), \min(0.8, 0.4))$$

$$= Max(0.5, 0.8, 0.4) = 0.8$$

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 \textit{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 OODBs 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 \textit{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, \textit{message}, is received by the object.

All objects which have the same attributes and methods are grouped into a higher level object, called a \textit{class}. An object is called an \textit{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 \textit{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 \textit{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 *A1* 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 mathematics, 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 AI 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 a 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].
- *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 \((P, T, A, M_0)\) where:

i. \( P \) is a finite set of \textit{places}.

ii. \( T \) is a finite set of \textit{transitions}.

iii. \( A \subset (P \times T \cup T \times P) \) is a finite set of \textit{arcs} denoting connections between places and transitions.

iv. \( M_0: P \rightarrow \{0, 1, 2, \ldots\} \) is the \textit{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 \textit{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 or 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:

\[
M'(p) = \begin{cases} 
M(p) + 1, & \forall \, p \in \text{output places} \\
M(p) - 1, & \forall \, p \in \text{input places} \\
M(p), & \text{otherwise}
\end{cases}
\]  \hspace{1cm} (2.4)

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

1. \(P,T,A\) are the same as those in regular Petri nets.

   i. \(C\) 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. \(E:A \rightarrow \text{Expression } E(a) \text{ of type } C(p(a))\), is the arc function. It maps each arc to an expression called an arc expression.

   v. \(G:T \rightarrow \text{Boolean expression}\), 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* \( d_i \) *THEN* \( d_k \) *with Certainty Factor* \( \mu_i \)

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

vi. \( \alpha : P \rightarrow [0, 1] \) is a mapping to a truth value of a token.

vii. \( \beta : P \rightarrow \mathbb{D} \) 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 \( \lambda \) 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 \( \tau, (y_k = \tau(I(t_i), \mu_i), \text{where } y_k \text{ is } \alpha(O(t_i))) \).

**Example:** The fuzzy deductive rule:

\[ \text{IF } d_i \text{ THEN } d_k \text{ with Certainty Factor } \mu_i \]

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, \( y_k = y_i \ast \mu_i \).

### 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:

\[ R_1: \]

**UN 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.
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 \( R_T \subseteq R \). 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 \( d_{b_i} \) and triggered rule set \( R_T = \{ r_1, r_2 \} \) and the (unique) final state is characterized by a database state \( d_{b_f} \) 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 \( r_j \) followed by \( r_i \) produces the same rule execution state as executing \( r_i \) followed by \( r_j \); this is shown in Figure 2.8.

If all pairs of rules in a rule set \( R \) commute, any execution sequences with...
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, con-
straints 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 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 lan-
guages; i.e., languages that support only a limited set of constructs for specifying
active behavior. If the rules become complex (like having composite events, sup-
porting 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 (ac-
tive or deductive), unlike the previous works, which do not model active rules.
Second, we not only model the composition of conditions but also the composi-
tion 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 combina-
tion 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 parame-
ter 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:

$$\text{strength} < e, r, s > = \mu_s(r) \ast \mu_e(e)$$  \hspace{1cm} (3.1)

which uses scalar multiplication, where $value(e_c)$ is the value of the event (either fuzzy or crisp) detected, $\mu_e$ 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 prior 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:

$$< \text{IN event list } \text{event threshold} >$$

$$\text{IF condition list } \text{EC coupling}$$

$$\text{THEN action/conclusion } \text{CA coupling}$$
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

\[ R_1: \text{ON } x_1 \text{ is } A_{11} \text{ AND/OR } x_2 \text{ is } A_{12} < \text{event.threshold}_{R_1} > \]
\[ \text{IF } y_1 \text{ is } B_{11} \text{ AND/OR } y_2 \text{ is } B_{12} \]
\[ \text{THEN } z \text{ is } C_1 \]

\[ R_2: \text{ON } x_1 \text{ is } A_{21} \text{ AND/OR } x_2 \text{ is } A_{22} < \text{event.threshold}_{R_2} > \]
\[ \text{IF } y_1 \text{ is } B_{21} \text{ AND/OR } y_2 \text{ is } B_{22} \]
\[ \text{THEN } z \text{ is } C_2 \]

\[ \ldots \]

\[ R_n: \text{ON } x_1 \text{ is } A_{n1} \text{ AND/OR } x_2 \text{ is } A_{n2} < \text{event.threshold}_{R_n} > \]
\[ \text{IF } y_1 \text{ is } B_{n1} \text{ AND/OR } y_2 \text{ is } B_{n2} \]
\[ \text{THEN } z \text{ is } C_n \]

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 \(\mu_{e_1}(value(e_c))\) in order to determine the membership value of the fuzzy event detected. For that either min (for conjunc-
tions of events) or max (for disjunctions of events) operator is used

\[ \text{strength} < e, r, s > = \mu_s(r) \ast \mu_{ef}(value(e_c)) \]

where

\[ \mu_{ef}(value(e_c)) = \mu_{A_{11} \text{and/or} A_{12}}(x_{10}, x_{20}) = \min/\max(\mu_{A_{11}}(x_{10}), \mu_{A_{12}}(x_{20})) \]  \hspace{1cm} (3.2) \]

If \( \text{strength} < e, r, s > \geq \text{event\_threshold}_r_i \), 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 conjunction of conditions and \( \max \) operator for the disjunction of conditions.

\[ \mu_{B_{11} \text{and/or} B_{12}}(y_{10}, y_{20}) = \min/\max(\mu_{B_{11}}(y_{10}), \mu_{B_{12}}(y_{20})) \]  \hspace{1cm} (3.3) \]

If \( \mu_{B_{11} \text{and/or} B_{12}}(y_{10}, y_{20}) > 0 \), 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.

\[ \mu_{A_{ri}} = \text{product}(\text{strength} < e, r, s >, \mu_{B_{11} \text{and/or} B_{12}}(y_{10}, y_{20})) \]  \hspace{1cm} (3.4) \]

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

\[ \mu_{C_{i}'}(z) = \min(\mu_{A_{ri}}, \mu_{C_i}(z)) \]  \hspace{1cm} (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.

\[ \mu_{C'}(z) = \max(\mu_{C_{i}'}(z), \ldots, \mu_{C_{n}'}(z)) \]  \hspace{1cm} (3.6) \]
3.3 A model for the Similarity of the Rule to the Scenario

Any active rule can be written as:

\[ C : \neg A_1, A_2, \ldots, A_n \]  

(3.7)

which is equivalent to

\[ A_1 \land A_2 \land \ldots \land A_n \rightarrow C \]  

(3.8)

where antecedent \((A_i\text{'s})\) 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 \(S\):

\[ \mu_s(r) = \max(\min(\max(\mu(A_s, A_r)), \max(\mu(C_s, C_r)))) \ast RLV_{rs} / RLV_{max} \]  

(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:

\begin{itemize}
  \item \textbf{R1:}
  \item JN there is a change on any motor temperature
    \item AND motor.temperature is hot 0.1
    \item IF some motors hot AND most deitas big_positive
\end{itemize}
THEN alarmnotification is low

R2:

JN 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:

JN 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:

JN 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:

JN 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:

JN 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):

\textbf{Remergency1:}
\begin{quote}
\textbf{JN} there is a change on any motor temperature \\
\textbf{AND} motor.temperature is hot \\
\textbf{IF} most motors hot \textbf{AND} most \text{delta}s big\_positive \\
\textbf{THEN} alarm\text{notification} is high
\end{quote}

\textbf{Remergency2:}
\begin{quote}
\textbf{JN} there is a change on any motor temperature \\
\textbf{AND} motor.temperature is hot \\
\textbf{IF} most motors very\_hot \\
\textbf{THEN} alarm\text{notification} is high
\end{quote}

According to the above meta rules, emergency scenario is $R_{\text{emergency1}} \lor R_{\text{emergency2}}$. This means that any rule similar to either $R_{\text{emergency1}}$ or $R_{\text{emergency2}}$ 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°C and 160°C and at time $t_2$ they are measured as 125°C and 160°C respectively. Assume also that for the active rule evaluation following membership functions and similarity relations are used:

\textit{temperature:}
\begin{itemize}
    \item normal trapezoidal(0,0,120,140)
    \item hot trapezoidal(120,140,300,300)
    \item very\_hot trapezoidal(145,160,300,300)
\end{itemize}

\textit{delta:}
\begin{itemize}
    \item negative trapezoidal(-1,-1,0.4,-0.2)
    \item big\_negative trapezoidal(-1,-1,0.8,-0.6)
    \item small\_negative trapezoidal(-0.8,-0.6,-0.4,-0.2)
\end{itemize}
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

<table>
<thead>
<tr>
<th>Temperature</th>
<th>normal</th>
<th>hot</th>
<th>very_hot</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>1</td>
<td>0.4</td>
<td>0</td>
</tr>
<tr>
<td>hot</td>
<td>0.4</td>
<td>1</td>
<td>0.9</td>
</tr>
<tr>
<td>very_hot</td>
<td>0</td>
<td>0.9</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.2: Similarity Matrix for AlarmNotification

<table>
<thead>
<tr>
<th>AlarmNotification</th>
<th>zero</th>
<th>low</th>
<th>medium</th>
<th>high</th>
</tr>
</thead>
<tbody>
<tr>
<td>zero</td>
<td>1</td>
<td>0.7</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>low</td>
<td>0.7</td>
<td>1</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>medium</td>
<td>0.5</td>
<td>0.7</td>
<td>1</td>
<td>0.7</td>
</tr>
<tr>
<td>high</td>
<td>0</td>
<td>0.5</td>
<td>0.7</td>
<td>1</td>
</tr>
</tbody>
</table>

Fuzzy Event Matching Phase:

Fuzzify the event:

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

$RLV(R_{\text{emergency}1, \text{Emergency}}) = 1$, $RLV(R_{\text{emergency}2, \text{Emergency}}) = 1.5$

Using the similarity matrices, emergency scenario meta-rules and relevance values, $\mu_{\text{emergency}}(r_1)$, i.e. relatedness of $R_1$ to the emergency scenario, is calculated as follows:

$$\text{Max} [\text{Min} (\mu_{\text{sim}}(\text{hot},\text{hot}), \mu_{\text{sim}}(\text{most},\text{some}), \mu_{\text{sim}}(\text{most},\text{most}), \mu_{\text{sim}}(\text{big positive},\text{big positive}), \mu_{\text{sim}}(\text{high},\text{low})) * 1/1.5, \mu_{\text{sim}}(\text{high},\text{low}) * 1.5/1.5]$$

$$\text{Min} (1, 0.5, 1, 1, 0.5) * 1/1.5.$$ 

$$\text{Max} (1, 0.5, 0.9, 0.5) * 1.5/1.5 =$$

$$\text{Max} (0.5 * 1/1.5, 0.5 * 1.5/1.5) =$$

$$\text{Max} (0.33, 0.5) = 0.5$$

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

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

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

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

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

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

for $R_2$ : $0.5 * \mu_{\text{hot}}(125) = 0.5 * 0.25 = 0.125$
for $R_3 : 0.5 \times \mu_{hot}(125) = 0.5 \times 0.25 = 0.125$
for $R_4 : 0.7 \times \mu_{hot}(125) = 0.7 \times 0.25 = 0.175$
for $R_5 : 0.5 \times \mu_{hot}(125) = 0.9 \times 0.25 = 0.225$
for $R_6 : 1.0 \times \mu_{hot}(125) = 1.0 \times 0.25 = 0.25$
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 $\mu_{biq_positive}(1) = 1$ and $\mu_{biq_positive}(0) = 0$. 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 $\mu_{biq_positive}(1) = 1$ and $\mu_{biq_positive}(0) = 0$. 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 $\mu_{biq_positive}(1) = 1$ and $\mu_{biq_positive}(0) = 0$. 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.
i. 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: $T \to TF$ is transition type function, mapping each transition $\in T$ to a transition function $\in TF$.

v. $A \subseteq (P \times T \cup T \times P)$ is a finite set of arcs for connections between places and transitions, where

   - $I: P \to T$ is an input mapping.
   - $U: T \to P$ 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: $P \to TT$ 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: $Arc \to expression$, 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 ($\mu_t$) 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 $TV(p_i) = \mu_i$, $\mu_i \in [0,1]$ and $PPM(p_i) = d_i$. This states that the degree of the truth of proposition $d_i$ is $\mu_i$. A transition $t_i$ is enabled if $\forall p_i \in I(t_i)$, $\mu_i > 0$. If this transition $t_i$ is fired,
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. $\mu_k = TV(p_k)$, is part of the token and gets calculated within the transition function $TF_i$, where $\mu_k = TF_i(I(t_i))$

Example: The fuzzy deductive rule \textit{IF} $d_i$ and $d_i$ \textit{THEN} $d_k$ can be modeled as shown in Figure 4.1. In this example, $PPM(p_i) = d_i$, $PPM(p_i) = d_i$, $PPM(p_k) = d_k$, $TV(p_i) = \mu_i$ and $TV(p_i) = \mu_i$. Suppose that $\mu_i > 0$ and $\mu_i > 0$. 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 $TV(p_k) = TF_n(I(t_n)) = \mu_i \cdot \mu_j = \mu_k$

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:

\[ \text{IF } e_1 < \text{event threshold} > \] 
\[ \text{IF } e_1 \] 
\[ \text{THEN } e_1 \]
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=$\{pr_1, pr_2, \ldots, pr_n\}$ 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 $P_c$.
   
   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. \( P_e \subseteq P \) is a finite set of output places, which correspond to fuzzified actions/conclusions. They are the final places in the fuzzy inference mechanism.

3. \( T = \{ t_1, t_2, \ldots, t_m \} \) 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. \( A \subseteq (P \times T \cup T \times P) \) is a finite set of arcs for connections between places and transitions. Connections between the input places and transitions \((P \times T)\) and connections between the transitions and output places \((T \times P)\) 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. \( AEF: A \rightarrow \text{expression}, \) is a mapping for each arc to an expression, which can be one of the following
a. an input arc expression for arcs $\in P \times I$, 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 $\in T \times P$ consisting of variables to be
assigned to output places

5. $TV : P \rightarrow [0, 1]$ is a mapping for the truth values of propositions ($\mu$) as-
signed 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 $\in I'$ to a token
type $\in TT$.

Figure 4.2 shows how we realize the steps of fuzzy inference using the FPN struc-
ture.

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

Finally, transitions are constructed over these FPN places. The pseudocode 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_strengthenedevent, 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
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:

\[ \text{UN there is a change in any motor temperature} \]
\[ \text{AND motor temperature is normal 0.2,} \]
\[ \text{IF delta is small positive} \]
\[ \text{THEN alarm notification is low.} \]

R2:

\[ \text{UN there is a change in any motor temperature} \]
AND motor temperature is not 0.2, 
IF delta is positive
THEN alarm notification is medium.

R3: 
UN 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: 
UN 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 t1 the measured temperature of the motor was 77°C and it has increased to 132°C at time t3.

Steps of the Inference Mechanism

1. Fuzzify the event
   \[\mu_{\text{normal}}(132) = 0.4, \mu_{\text{hot}}(132) = 0.6, \mu_{\text{veryhot}}(132) = 0\]

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.
   \[\mu_{\text{emergency}}(R_1) = 0.6, \mu_{\text{emergency}}(R_2) = 0.7, \mu_{\text{emergency}}(R_3) = 1\]
   and using the formula (1), event strengths are calculated as:
   for \(R_1\) : \(0.6 \times \mu_{\text{normal}}(132) = 0.6 \times 0.4 = 0.24\)
   for \(R_2\) : \(0.7 \times \mu_{\text{hot}}(132) = 0.7 \times 0.6 = 0.42\)
   for \(R_3\) : \(1 \times \mu_{\text{veryhot}}(132) = 1 \times 0 = 0\)
   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)
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_{\text{small positive}}(0.7) = 0.5, \min(0.5) = 0.5 \Rightarrow \text{match factor} = 0.5 
   - for \( R_2 \): \( \mu_{\text{positive}}(0.7) = 1, \min(1) = 1 \Rightarrow \text{match factor} = 1 

4. **Find the clipping values**
   - For this, the condition match factor is multiplied by the strength of event
     - for \( R_1 \): \( 0.24 \times 0.5 = 0.12 \) (clipping value for the alarm level of low)
     - for \( R_2 \): \( 0.42 \times 1 = 0.42 \) (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:

\( R_1 \):

```
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).
Steps of Inference Mechanism:

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

1. Find the condition match factor
   
   for \( R_1 \): \( \mu_{\text{large}}(\text{laundry\_quantity}) = 0.4 \),
   
   \( \mu_{\text{hard}}(\text{laundry\_softness}) = 0.2 \),
   
   \( \min(0.4, 0.2) = 0.2 \Rightarrow \text{match factor} = 0.2 \)
   
   for \( R_2 \): \( \mu_{\text{medium}}(\text{laundry\_quantity}) = 0.6 \),
   
   \( \mu_{\text{normal, hard}}(\text{laundry\_softness}) = 0.8 \),
   
   \( \min(0.6, 0.8) = 0.6 \Rightarrow \text{match factor} = 0.6 \)
   
   for \( R_3 \): \( \mu_{\text{small}}(\text{laundry\_quantity}) = 0 \),
   
   \( \mu_{\text{soft}}(\text{laundry\_softness}) = 0 \),
   
   \( \min(0, 0) = 0 \Rightarrow R_3 \text{ fails} \)
   
   for \( R_4 \): \( \mu_{\text{medium}}(\text{laundry\_quantity}) = 0.5 \),
   
   \( \mu_{\text{soft}}(\text{laundry\_softness}) = 0 \),
   
   \( \min(0.5, 0) = 0 \Rightarrow R_4 \text{ fails} \)
   
   for \( R_5 \): \( \mu_{\text{large}}(\text{laundry\_quantity}) = 0.5 \),
   
   \( \mu_{\text{soft}}(\text{laundry\_softness}) = 0 \),
   
   \( \min(0.5, 0) = 0 \Rightarrow R_5 \text{ fails} \).
for $R_6$; $\mu_{\text{small}}(\text{laundry\_quantity}) = 0$.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$\min(0.6, 0.2) = 0.2 \Rightarrow$ 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 $\mu_{\text{event\_strength}}$ is always 1.

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

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

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

for $R_2$; $1 \times 0.2 = 0.2$ (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 */

define_true_cycles = determine_true_cycles()

if (NULL == (true_cycle_list = determine_true_cycles())) then
  Termination guaranteed:
else
  inform_user(true_cycle_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 $m \times m$ 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 $m \times m$ 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.
5.1.2 Example

Suppose there are the following two rules:

R1:
\[ \text{ON} \ (e_{11} \text{ and } e_{21}) \]
\[ \text{IF } c_1 \]
\[ \text{THEN } a_1 \]

R2:
\[ \text{ON} \ e_{12} \]
\[ \text{IF } c_2 \]
\[ \text{THEN } a_2 \]

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:

\[
D_1 = \begin{bmatrix}
1 & 1 \\
1 & 0
\end{bmatrix} \quad L_1 = \begin{bmatrix}
1,1 & 1,2 \\
2,1 & 0
\end{bmatrix}
\]

Since \( D_1[1][1] \) entry has the value 1, this means that there is a cycle. After checking \( r_1,PN\_primitive\_event\_rule\_set, \) rules \( r_1 \) and \( r_2 \) are found. Since the cyclic set \( \{1,1\} \) does not contain \( r_2 \), this is not a true cycle. Therefore it is excluded. Then \( D_2 \) is calculated and at the same time \( L_2 \) is obtained:
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 $((D_k \neq 0) \text{ and } (k \leq \text{number of rules}))$ 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 \( 0(\text{zero}) \) 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 \( 0(\text{zero}) \).

**Lemma 5.1.3.3:** Let rule \( r_i \) have a composite event. Although the \( D_n[i][i] \) entry is not \( 0(\text{zero}) \) 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 \( 0(\text{zero}) \) (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 \(1_R\)
Begin
for (each inference group \(inf_i\)) do
    for (each inference group \(inf_j\)) do
        if (\(\text{read\_set}(inf_i) == \text{write\_set}(inf_j)\)) then
            add \(inf_i\) and \(inf_j\) to \(1_R\)
        end if
    end for
end for
return \(1_R\)
End

Figure 5.5: Check\_Confluence Algorithm

the pseudocode 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 \(m\times1\) column vector of places \(p_i\), where \(i=1,\ldots,m\). Each of its entries holds a data structure of three elements:

- 1\(^{st}\) element holds the place id.
- 2\(^{nd}\) element holds the object id which uses the place,
- 3\(^{rd}\) element holds the token value.

b. \(current\_places\) is a linked list. Each of its entries holds a data structure of two elements:

- 1\(^{st}\) element holds the place id.
- 2\(^{nd}\) element holds the index of place id in \(M\) vector.
Begin
find the index of the start place $P_{st}$ 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 dc
                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 $c_1^+$,
                        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 $n \times 1$ column vector of transitions $t_i$, 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 $token\_values$.

1. $C^+ = (c^+_{ij})$ 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 occurrence.

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 occurrences 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

70
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 \textit{fuzzywinddr}, its crisp type be \textit{real}. It has a \textit{fuzzyOR} semantic, and it contains the fuzzy terms \textit{north}, \textit{northwest}, \textit{west}, \textit{southwest}, \textit{south}, \textit{southeast}, \textit{east} and \textit{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.
Figure 6.4: Fuzzy type definition

Figure 6.5: Similarity matrix 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
Figure 6.7: Class creation (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, \textit{when} should be \textit{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 \textit{when} should be \textit{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, \textit{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 \textit{wind\_value}, \textit{prev\_wind\_value} and \textit{wind\_ch\_vel} etc. In
this example, \texttt{prev\_wind\_value} is of a \textit{before} type derived attribute, which should be updated before the \texttt{wind\_value} gets updated. However, \texttt{wind\_ch\_vel} is of an \textit{after} type derived attribute, which should be updated after the \texttt{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
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
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
Figure 6.12: Rule operations (1)

Figure 6.13: Active Rule Definition - Rule operations (2)
Figure 6.14: Deductive Rule Definition - Rule operations (3)
Figure 6.16: Updating Current Scenario - Rule operations

Figure 6.17: Class browser
Figure 6.18: Object browser

Figure 6.19: Steps in Evaluation of a User Query
Figure 6.20: Simple Query

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: \( n \times (p_c + p_e) \).

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 \( n \times p_e \) 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 \( 2 \times g \times n_g \) number of operations. Possible number of operations in this part is: \( n^2 + n \times p_e + 2 \times g \times n_g \).

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 \( e_p \times n \times p_e \) operations. Then, for each fuzzy composite condition, we initialize related links. If we have \( c_c \) number of composite events, each having \( p_c \) number of component places at the worst case, this requires \( c_c \times p_c \) 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 fuzzyprimitiveevent 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 fuzzyprimitivecondition 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 ≤ 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 \( c_c \times p_c \) operations. For each rule, we create trigger condition transitions to the fuzzy primitive conditions of the rule. This takes \( n \times p_c \) 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 \( g \times n_q \) operations. In order to create event generating transitions, we check each fuzzy primitive condition with the primitive events and we check each fuzzy action with the primitive events. In worst case, this takes \( e_p \times (p_c + n) \) operations. Therefore, a possible number of operations in this part is:

\[
e_p \times n \times p_c + c_c \times p_c + c_c \times p_c + n \times p_c + g \times n_q + e_p \times (p_c + n)
\]

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 \( 2 \times n \times t \times p \) 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 $p_i$) 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 $p_i$ 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 $r_i$ in the $r_i$'s fuzzy inference group) do
    create obtain fuzzy action transition, update the links
  end for
end if
for (each place $p_i$ either a fuzzy action or fuzzy primitive condition) do
  for (each primitive event $p_i$) 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 introduced 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, anc
• 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>
<thead>
<tr>
<th>Scenario</th>
<th>Weather</th>
<th>Weather Event</th>
<th>Temperature Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summer</td>
<td>Clear, Clear few</td>
<td>-</td>
<td>Increase</td>
</tr>
<tr>
<td>Winter</td>
<td>Clouded stable</td>
<td>Rain, Fog, Snow</td>
<td>Decrease</td>
</tr>
<tr>
<td>Spring, Fall</td>
<td>Clouded instable</td>
<td>Rain, Hail, Shower</td>
<td>Increase</td>
</tr>
</tbody>
</table>

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 livelihood 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 Suijthjorn 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 occurring 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 algorithmic solution too complex.

7.4 Active Rules and FPN Construction

7.4.1 1\textsuperscript{st} 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(broksenk),
    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 UNSTABLE 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(cloudedunstable)

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 R₁ to R₄ use.

- If all changes occur, rules R₁ to R₄ 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:

**Step 1** Once we have the values of atmospheric elements, which take part in triggering the rules R₁ to R₄, 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 R₃ requires.

**Step 2** We perform the **composition of events**, which are fuzzified at Step 1. For that we use min for AND and max for OR operator and obtain the composite event.

**Step 3** Using the similarity of each rule and the composite events calculated at Step 2, we obtain the **event strengths** for each rule.

**Step 4** 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, R₃ checks whether the previous wind value was *breeze* or *medium strong*.

**Step 5** We perform the **composition of conditions**, which are fuzzified at Step 4. For that we use min for AND and max for OR operator and obtain the **composite condition**.

**Step 6** Using the event strengths calculated at Step 3 and composite conditions calculated at Step 5, we calculate the **clipping value** for each rule.

**Step 7** We **combine the clipping values** of all rules calculated at Step 6.

**Step 8** We find the **overall fuzzy action** within the combined actions in Step 7, 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 R₁ to R₄. Therefore, without rules R₁ to R₄ 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 R₅ to R₉ with a weather forecasted using rules R₁ to R₄

- 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 R₅ to R₉ use

- If all changes occur, rules R₅ to R₉ 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)
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

RO1: (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 \text{R01} is converted to the following active rule, which has an abstract kind of event

\textbf{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)

\subsection*{7.5.2 Rules for querying the weather event forecast}

\textbf{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(cloudeedinstable).
color(color(darkgrey):
wind.prev_wind_value(calm),
cloud.prev_color(grey)
THEN forecast.weather event(shower)

R72: (SNOW)
   IF forecast.weather(cloudeedstable),
wind.wind_dr(north).
color(color(grey)
temperature.temp_value(cold),
wind.prev_wind_dr(south),
color(color(white))
THEN forecast.weather event(snow)

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

R92: (FUG)
    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)
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 R₃, R₄, R₅, R₇, R₉, R₁₂, R₁₈, R₁₀₂, R₁₁₂, R₁₂₂. 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

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Place Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>pme</td>
<td>primitive event</td>
</tr>
<tr>
<td>fpe</td>
<td>fuzzy primitive event</td>
</tr>
<tr>
<td>fce</td>
<td>fuzzy composite event</td>
</tr>
<tr>
<td>rlp</td>
<td>rule place</td>
</tr>
<tr>
<td>ste</td>
<td>strengthened event</td>
</tr>
<tr>
<td>fpc</td>
<td>fuzzy primitive condition</td>
</tr>
<tr>
<td>fcc</td>
<td>fuzzy composite condition</td>
</tr>
<tr>
<td>cpa</td>
<td>clipped action</td>
</tr>
<tr>
<td>cma</td>
<td>combined action</td>
</tr>
<tr>
<td>fza</td>
<td>fuzzy action</td>
</tr>
</tbody>
</table>

Figure 7.7: Termination Analysis for the example rules
Figure 7.8: Confluence Analysis for the example rules

Figure 7.9: Petri Net (1)
Figure 7.10: Petri Net (2)

Figure 7.11: Petri Net (3)
Figure 7.12: Petri Net (4)

Figure 7.13: Petri Net (5)
Figure 7.14: Petri Net (6)

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Sensor Values</th>
<th>Fuzzification Results for event / condition evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>at time ( t_1 )</td>
<td>( R_3 ) \hspace{1cm} ( R_4 )</td>
</tr>
<tr>
<td>Wind Value</td>
<td>6 \hspace{1cm} 6.5</td>
<td>0.5 / - \hspace{1cm} - / -</td>
</tr>
<tr>
<td>Cloud Orientation</td>
<td>0 \hspace{1cm} 135</td>
<td>0.5 / - \hspace{1cm} 0.5 / -</td>
</tr>
<tr>
<td>Cloud Base</td>
<td>7200 \hspace{1cm} 1600</td>
<td>1 / - \hspace{1cm} 1 / 0.5</td>
</tr>
<tr>
<td>Wind Direction</td>
<td>255 \hspace{1cm} 165</td>
<td>- / - \hspace{1cm} - / 0.66</td>
</tr>
<tr>
<td>Humidity</td>
<td>50 \hspace{1cm} 57</td>
<td>1 / - \hspace{1cm} 1 / 0.5</td>
</tr>
<tr>
<td>Temperature value</td>
<td>-4.5 \hspace{1cm} -1</td>
<td>1 / - \hspace{1cm} 1 / 0.5</td>
</tr>
<tr>
<td>Pressure value</td>
<td>1003.5 \hspace{1cm} 1000</td>
<td>1 / - \hspace{1cm} 1 / 0.5</td>
</tr>
</tbody>
</table>

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_{\text{pressure\_change\_direction\_decreasing}}(-3.5) = 1. \\
   \mu_{\text{humidity\_change\_direction\_increasing}}(+7) = 1. \\
   \mu_{\text{wind\_value\_medium\_strong\_strong}}(6.5) = 0.5 \\
   \mu_{\text{cloud\_orientation\_vertical}}(135) = 0.5 \\
   \mu_{\text{cloud\_base\_change\_direction\_decreasing}}(-5600) = 1. \\
   \mu_{\text{temperature\_change\_direction\_increasing}}(+3.5) = 1. \\
   \min(1, 1, 0.5, 0.5, 1, 1) = 0.5. \text{ Since } 0.5 > 0, R_3 \text{ gets fired.}
   
   Evaluate \( R_4 \):
\[ \mu_{\text{pressure change direction decreasing}}(-3.5) = 1. \]
\[ \mu_{\text{humidity change direction increasing}}(+7) = 1. \]
\[ \mu_{\text{temperature change direction increasing}}(+3.5) = 1. \]
\[ \mu_{\text{cloud base change direction decreasing}}(-5600) = 1. \]
\[ \mu_{\text{cloud orientation horizontal}}(135) = 0.5. \]
\[ \min(1, 1, 1, 1, 0.5) = 0.5. \] Since \( 0.5 > 0 \), \( R_4 \) gets fired

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

For this \( \mu_{\text{winter}}(r_i) \) values for the rules are calculated using the Formula 3.9.
\[ \mu_{\text{winter}}(R_3) = 0, \mu_{\text{winter}}(R_4) = 1 \]
and using the Equation 3.1, event strengths are calculated as:
for \( R_3 \) : \( 0 \times 0.5 = 0. \)
for \( R_4 \) : \( 1 \times 0.5 = 0.5 \)
Threshold values for the rules are given as 0.1, therefore only \( R_4 \) succeed this step.

3. Find the Condition Match Factor

Evaluate \( R_4 \).
\[ \mu_{\text{pressure change velocity slow}}(-3.5) = 0.5. \]
\[ \mu_{\text{humidity change velocity slow}}(+7) = 0.5. \]
\[ \mu_{\text{temperature change velocity slow}}(+3.5) = 0.5. \]
\[ \mu_{\text{cloud base change velocity slow}}(-5600) = 0.5. \]
\[ \mu_{\text{wind direction southeast/south}}(210) = 0.66. \]
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
\( 0.5 \times 0.5 = 0.25. \)
So clipping value of 0.25 is used for the action part of \( R_4 \), which is an forecasted weather of cloudy/stable.

5. Find the fuzzy action

Maximum of the clipping values are taken, since we have \( R_4 \), \( \max(0.25) = 0.25 \).
Figure 7.16: FPN used for forecasting the weather

0.25. This means a weather forecast of cloudy.stable.

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 dynamics(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 $R_{33}$**:

   $$\min(1, 1, 0.5, 0.5, 1, 1, 0.5, 0.2, 1) = 0.$$  \text{fails}

   **Evaluate $R_{43}$**: 

   121
min(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 \( \mu_{\text{eventstrength}} \) is always 1 for \( R_{42}; 1 \times 0.5 = 0.5 \) (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 $R_5$.

   $\mu_{\text{expected weather clouded stable}} = 0.25$.

   $\mu_{\text{wind direction south/southwest}(22.5)} = 0$.

   $\mu_{\text{wind value medium strong}(6)} = 0$.

   $\mu_{\text{cloud color arey}(4)} = 0.5$.

   $\min(0.25, 0, 0, 0.5) = 0$. $R_5$ fails.

   Evaluate $R_7$.

   $\mu_{\text{expected weather clouded stable}} = 0.25$.

   $\mu_{\text{wind direction north}(337.5)} = 0.5$.

   $\mu_{\text{cloud color arey}(4)} = 0.5$.

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

   Evaluate $R_6$.

   $\mu_{\text{expected weather clouded stable}} = 0.25$.

   $\min(0.25) = 0.25$. Since $0.25 > 0$ $R_6$ 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.

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

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

- for $R_7$ : $1 \times 0.25 = 0.25$
- for $R_9$ : $1 \times 0.25 = 0.25$

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

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

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 $R_7$, $0.25 \times 0.5 = 0.125$
   
   for $R_9$, $0.25 \times 1 = 0.25$
   
   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 $R_9$, $\text{max}(0.125,0.25) = 0.25$. 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 \( R_{112} \):**

   \[
   \mu_{\text{expected_weather_clear/clear_few/clouded_instable}} = 0.5 \\
   \mu_{\text{pressure_change_direction_decreasing}(3.5)} = 0. \\
   \mu_{\text{wind_direction_south/southwest}(337.5)} = 0. \\
   \mu_{\text{prev_wind_direction_north/northwest/northeast}(337.5)} = 0. \\
   \min(0.5, 0, 0, 0) = 0, \ R_{112} \text{ fails.}
   \]

   **Evaluate \( R_{119} \):**

   \[
   \mu_{\text{expected_weather_clouded_stable}} = 1 \\
   \mu_{\text{pressure_change_direction_increasing}(3.5)} = 1. \\
   \mu_{\text{wind_direction_north/northwest/northeast}(337.5)} = 0.5. \\
   \mu_{\text{prev_wind_direction_south/southwest}(337.5)} = 0.67. \\
   \min(1, 1, 0.5, 0.67) = 0.5. \text{ Since } 0.5 > 0 \ R_{112} \text{ gets fired.}
   \]

   **Evaluate \( R_{199} \):**

   \[
   \mu_{\text{expected_weather_clear/clear_few/clouded_instable/clouded_stable}} = 1 \\
   \mu_{\text{pressure_change_direction_nochange}(3.5)} = 0.
   \]
\[ \mu_{wind\_direction\_west}(337.5) = 0. \]
\[ \mu_{wind\_value\_breeze}(6) = 1. \]
\[ \min(1,0,0,1) = 0. \text{ R}_{122} \text{ fails} \]

2. **Find the clipping values**

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

For the abstract kind of events \( \mu_{event\_strength} \) is always 1

for \( R_{112} \): \( 1 \times 0.5 = 0.5 \) (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

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.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Composition of events?</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Uncertainty/ Fuzziness?</td>
<td>Yes, No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Is parameter passing available?</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Rule partitioning? (scenario)</td>
<td>No</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>Yes</td>
</tr>
<tr>
<td>Different Fuzzy Terms in Events of Concurrent Rules?</td>
<td>No</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>Yes</td>
</tr>
<tr>
<td>Object–oriented?</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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

Figure 8.2: Fuzzy Petri Nets: A comparison.

...environmental 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 implementation 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


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


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

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>trapezoidal</td>
<td>992 992 1008 1024</td>
</tr>
<tr>
<td>high</td>
<td>trapezoidal</td>
<td>1008 1024 1040 1040</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>nochange</td>
<td>trapezoidal</td>
<td>0 0 1 2</td>
</tr>
<tr>
<td>slow</td>
<td>trapezoidal</td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>fast</td>
<td>trapezoidal</td>
<td>3 4 5 5</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>decreasing</td>
<td>trapezoidal</td>
<td>-5 -5 -3 -1</td>
</tr>
<tr>
<td>nochange</td>
<td>trapezoidal</td>
<td>-3 -1 1 3</td>
</tr>
<tr>
<td>increasing</td>
<td>trapezoidal</td>
<td>1 3 5 5</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>cold</td>
<td>trapezoidal</td>
<td>-5 -5 5 10</td>
</tr>
<tr>
<td>chill</td>
<td>trapezoidal</td>
<td>5 10 15 20</td>
</tr>
<tr>
<td>warm</td>
<td>trapezoidal</td>
<td>15 20 25 30</td>
</tr>
<tr>
<td>hot</td>
<td>trapezoidal</td>
<td>25 30 35 35</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>nochange</td>
<td>trapezoidal</td>
<td>0 0 1 2</td>
</tr>
<tr>
<td>slow</td>
<td>trapezoidal</td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>fast</td>
<td>trapezoidal</td>
<td>3 4 5 5</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>decreasing</td>
<td>trapezoidal</td>
<td>-5 -5 -3 -1</td>
</tr>
<tr>
<td>nochange</td>
<td>trapezoidal</td>
<td>-3 -1 1 3</td>
</tr>
<tr>
<td>increasing</td>
<td>trapezoidal</td>
<td>1 3 5 5</td>
</tr>
</tbody>
</table>
Table A.7: Membership Functions for **humidity value** attribute

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>trapezoidal</td>
<td>0 0 20 40</td>
</tr>
<tr>
<td>ideal</td>
<td>trapezoidal</td>
<td>20 40 60 80</td>
</tr>
<tr>
<td>high</td>
<td>trapezoidal</td>
<td>60 80 100 100</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>nochange</td>
<td>trapezoidal</td>
<td>0 0 2 4</td>
</tr>
<tr>
<td>slow</td>
<td>trapezoidal</td>
<td>2 4 6 8</td>
</tr>
<tr>
<td>fast</td>
<td>trapezoidal</td>
<td>6 8 10 10</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>decreasing</td>
<td>trapezoidal</td>
<td>-10 -10 -6 -2</td>
</tr>
<tr>
<td>nochange</td>
<td>trapezoidal</td>
<td>-6 -2 2 6</td>
</tr>
<tr>
<td>increasing</td>
<td>trapezoidal</td>
<td>2 6 10 10</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>calm</td>
<td>trapezoidal</td>
<td>0 0 3 4</td>
</tr>
<tr>
<td>breeze</td>
<td>trapezoidal</td>
<td>3 4 6 7</td>
</tr>
<tr>
<td>medium_strong</td>
<td>trapezoidal</td>
<td>6 7 15 16</td>
</tr>
<tr>
<td>strong</td>
<td>trapezoidal</td>
<td>15 16 30 31</td>
</tr>
<tr>
<td>storm</td>
<td>trapezoidal</td>
<td>30 31 50 50</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>nochange</td>
<td>trapezoidal</td>
<td>0 0 2 4</td>
</tr>
<tr>
<td>slow</td>
<td>trapezoidal</td>
<td>2 4 6 8</td>
</tr>
<tr>
<td>fast</td>
<td>trapezoidal</td>
<td>6 8 10 10</td>
</tr>
</tbody>
</table>
Table A.12: Membership Functions for *wind direction* attribute

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>north</td>
<td>triangular</td>
<td>-45</td>
</tr>
<tr>
<td>northwest</td>
<td>triangular</td>
<td>0</td>
</tr>
<tr>
<td>west</td>
<td>triangular</td>
<td>45</td>
</tr>
<tr>
<td>southwest</td>
<td>triangular</td>
<td>90</td>
</tr>
<tr>
<td>south</td>
<td>triangular</td>
<td>135</td>
</tr>
<tr>
<td>southeast</td>
<td>triangular</td>
<td>180</td>
</tr>
<tr>
<td>east</td>
<td>triangular</td>
<td>225</td>
</tr>
<tr>
<td>northeast</td>
<td>triangular</td>
<td>270</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>nochange</td>
<td>trapezoidal</td>
<td>0</td>
</tr>
<tr>
<td>slow</td>
<td>trapezoidal</td>
<td>18</td>
</tr>
<tr>
<td>fast</td>
<td>trapezoidal</td>
<td>54</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear</td>
<td>triangular</td>
<td>0</td>
</tr>
<tr>
<td>few</td>
<td>trapezoidal</td>
<td>0</td>
</tr>
<tr>
<td>broken_sky</td>
<td>trapezoidal</td>
<td>2</td>
</tr>
<tr>
<td>cloudy</td>
<td>trapezoidal</td>
<td>4</td>
</tr>
<tr>
<td>overcast</td>
<td>trapezoidal</td>
<td>6</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>trapezoidal</td>
<td>0</td>
</tr>
<tr>
<td>medium</td>
<td>trapezoidal</td>
<td>2000</td>
</tr>
<tr>
<td>high</td>
<td>trapezoidal</td>
<td>5000</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>decreasing</td>
<td>trapezoidal</td>
<td>-8000</td>
</tr>
<tr>
<td>nochange</td>
<td>trapezoidal</td>
<td>-4800</td>
</tr>
<tr>
<td>increasing</td>
<td>trapezoidal</td>
<td>1600</td>
</tr>
</tbody>
</table>
Table A.17: Membership Functions for *cloud base change velocity* attribute

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>nochange</td>
<td>trapezoidal</td>
<td>0</td>
</tr>
<tr>
<td>slow</td>
<td>trapezoidal</td>
<td>1600</td>
</tr>
<tr>
<td>fast</td>
<td>trapezoidal</td>
<td>4800</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>fuzzy term</th>
<th>function</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>vertical</td>
<td>triangular</td>
<td>45 90 145</td>
</tr>
<tr>
<td>horizontal</td>
<td>triangular</td>
<td>-45 0 45</td>
</tr>
</tbody>
</table>
APPENDIX B

APPLICATION LOG FOR THE EXAMPLES

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

INFEERENCE 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 pb, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place pb
  Object id: cb, Object attr value: -5600.000000, Domain: fuzzycbasechdir,
  Fuzzytermlist: decreasing, Membership: 1.00

CURRENT PLACES FOR USE: p93 p94 p95 p96 p98 pb
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: cb, Object attr value: 135.000000, Domain: fuzzycorrient,
  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: 2

146
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: cb, Object attr value: 135.000000, Domain: fuzzycolor,
Fuzzytermist: 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: cb, Object attr value: 7.000000, Domain: fuzzynumchdir,
Fuzzytermist: 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: cb, Object attr value: -3.600000, Domain: fuzzypreschdir,
Fuzzytermist: 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: cb, Object attr value: 3.500000, Domain: fuzzytemphdir,
Fuzzytermist: 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.

147
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: cb, Object attr value: 6.500000, Domain: fuzzywind,
   Fuzzyterm list: medium strong 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 INERENCE 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 INPUT place p0, which is added to the tail of
CURRENT places list

TOKEN VALUE put in INPUT place p0

   Object id: cb, 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 INPUT place p13, which is added to the tail of
CURRENT places list

TOKEN VALUE put in INPUT place p13

   Object id: cb, 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: cb, Membership: 0.50
CURRENT PLACES FOR USE: p105
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: cb, Object attr value: 3.500000, Domain: fuzzypreschvel,
  Fuzzytermiist: 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: cb, Object attr value: 7.000000, Domain: fuzzymumchvel,
  Fuzzytermiist: 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: cb, Object attr value: 3.500000, Domain: fuzzymumchvel,
  Fuzzytermiist: 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: cb, Object attr value: 5600.000000, Domain: fuzzycbasechvel,
  Fuzzytermiist: 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: cb, Object attr value: 16b.0000000, Domain: fuzzywinddr,
  Fuzzytermiist: 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: cb, 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 p17
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: p17
PLACE FOR USE NOW: p17, 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: cb, Membership: 0.2b

CURRENT PLACES FOR USE: p22
PLACE FOR USE NOW: p22, TRANSITION FOR USE NOW: 90
I am in TRANSITION 90 now. which does CUMULATION.
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: cb, Fuzzytermist: cloudedstable, Clipping value: 0.2b

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: cb, Fuzzytermist: cloudedstable, Overall Clipping value: 0.2b

CURRENT PLACES FOR USE: p101
PLACE FOR USE NOW: p101, TRANSITION FOR USE NOW: 102
I am in TRANSITION 102 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: cb, Membership: 0.2b
B.2 Querying the Weather Forecast - Deductive Rule Example (1)

INFERENCES 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: cb, Object attr value: -3.500000, Domain: fuzziypreschdir,
   Fuzzytermist: 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: cb, Object attr value: 7.000000, Domain: fuzzyhumchdir,
   Fuzzytermist: 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: cb, Object attr value: 6.500000, Domain: fuzzywind,
   Fuzzytermist: 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 db2
Object id: cb, Object attr value: 135.000000, Domain: fuzzycorent,
Fuzzytermiist: 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 db3, which is added to the tail of
CURRENT places list

TOKEN VALUE put in OUTPUT place db3
Object id: cb, Object attr value: -5500.000000, Domain: fuzzybasechdir,
Fuzzytermiist: decreasing, 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: 29
I am in TRANSITION 29 now, which does TRIGGERING condition.
I mark the OUTPUT place db4, which is added to the tail of
CURRENT places list

TOKEN VALUE put in OUTPUT place db4
Object id: cb, Object attr value: 3.500000, Domain: fuzzytempchdir,
Fuzzytermiist: 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 db5, which is added to the tail of
CURRENT places list

TOKEN VALUE put in OUTPUT place db5
Object id: cb, Object attr value: 3.500000, Domain: fuzzypreschvei,
Fuzzytermiist: 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 db6, which is added to the tail of CURRENT places list

TOKEN VALUE put in OUTPUT place db6
Object id: cb, Object attr value: 7.000000, Domain: fuzzyhumchvei,
Fuzzytermiist: fast, Membership1: 0.50, Membership2: 1.00

CURRENT PLACES FOR USE: d90 d49 d50 d61 d52 d53 d54 d55 d56 d57 d68
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: cb, Object attr value: 5.000000, Domain: fuzzywind,
Fuzzytermiist: breeze mediumstrong, Membership1: 0.00, Membership2: 1.00

CURRENT PLACES FOR USE: d90 d49 d50 d61 d52 d53 d54 d55 d56 d57 d68
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: cb, Object attr value: -3.500000, Domain: fuzzypreschdir,
Fuzzytermiist: decreasing, Membership1: 1.00, Membership2: 1.00

CURRENT PLACES FOR USE: d90 d49 d50 d61 d52 d53 d54 d55 d56 d57 d68 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: cb, Object attr value: 7.000000, Domain: fuzzyhumchdir,
Fuzzytermiist: increasing, Membership1: 1.00, Membership2: 1.00

CURRENT PLACES FOR USE: d90 d49 d50 d61 d52 d53 d54 d55 d56 d57 d68 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: cb, Object attr value: 3.500000, Domain: fuzzytempchdir,
Fuzzytermiist: increasing, Membership1: 1.00, Membership2: 1.00

155
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: cb, Object attr value: -5600.000000, Domain: fuzzycbasechdir,
Fuzzytermiist: 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: cb, Object attr value: 135.000000, Domain: fuzzyccorient,
Fuzzytermiist: 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: cb, Object attr value: 3.500000, Domain: fuzzypreschvel,
Fuzzytermiist: 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: cb, Object attr value: 7.000000, Domain: fuzzyhumchvel,
Fuzzytermiist: 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 NEW: d90, TRANSITION FOR USE NEW: 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: cb, Object attr value: 3.000000, Domain: fuzzytempchvei,
  Fuzzytermiist: slow, Membership1: 0.50, Membership2: 1.0C

PLACE FOR USE NEW: d90, TRANSITION FOR USE NEW: 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: cb, Object attr value: b600.000000, Domain: fuzzycbasechvei,
  Fuzzytermiist: slow, Membership1: 0.50, Membership2: 1.0C

PLACE FOR USE NEW: d90, TRANSITION FOR USE NEW: 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: cb, Object attr value: 165.000000, Domain: fuzzywinddr,
  Fuzzytermiist: southeast south, Membership1: 0.67, Membership2: 1.0C

PLACE FOR USE NEW: d49, TRANSITION FOR USE NEW: 51
I am in TRANSITION 51 now, which does COMPOSITION.
Since composition result is 0, rule r32 is removed from the inference.

PLACE FOR USE NEW: d49, TRANSITION FOR USE NEW: 51
I am in TRANSITION 51 now, which does COMPOSITION.
Since composition result is 0, rule r32 is removed from the inference.
PLACE FOR USE NOW: d50. TRANSITION FOR USE NOW: b1
Since transition b1 is consumed before, do not fire it again.

CURRENT PLACES FOR USE: d51 d52 d53 d54 d55 d56 d57 d58 d59 d60 d61 d62 d63 d64 d65 d66 d67 d68 d69
PLACE FOR USE NOW: d51. TRANSITION FOR USE NOW: b1
Since transition b1 is consumed before, do not fire it again.

CURRENT PLACES FOR USE: d52 d53 d54 d55 d56 d57 d58 d59 d60 d61 d62 d63 d64 d65 d66 d67 d68 d69
PLACE FOR USE NOW: d52. TRANSITION FOR USE NOW: b1
Since transition b1 is consumed before, do not fire it again.

CURRENT PLACES FOR USE: d53 d54 d55 d56 d57 d58 d59 d60 d61 d62 d63 d64 d65 d66 d67 d68 d69
PLACE FOR USE NOW: d53. TRANSITION FOR USE NOW: b1
Since transition b1 is consumed before, do not fire it again.

CURRENT PLACES FOR USE: d54 d55 d56 d57 d58 d59 d60 d61 d62 d63 d64 d65 d66 d67 d68 d69
PLACE FOR USE NOW: d54. TRANSITION FOR USE NOW: b1
Since transition b1 is consumed before, do not fire it again.

CURRENT PLACES FOR USE: d55 d56 d57 d58 d59 d60 d61 d62 d63 d64 d65 d66 d67 d68 d69
PLACE FOR USE NOW: d55. TRANSITION FOR USE NOW: b1
Since transition b1 is consumed before, do not fire it again.

CURRENT PLACES FOR USE: d56 d57 d58 d59 d60 d61 d62 d63 d64 d65 d66 d67 d68 d69
PLACE FOR USE NOW: d56. TRANSITION FOR USE NOW: b1
Since transition b1 is consumed before, do not fire it again.

CURRENT PLACES FOR USE: d57 d58 d59 d60 d61 d62 d63 d64 d65 d66 d67 d68 d69
PLACE FOR USE NOW: d57. TRANSITION FOR USE NOW: b1
Since transition b1 is consumed before, do not fire it again.

CURRENT PLACES FOR USE: d58 d59 d60 d61 d62 d63 d64 d65 d66 d67 d68 d69
PLACE FOR USE NOW: d58. TRANSITION FOR USE NOW: b1
Since transition b1 is consumed before, do not fire it again.

PLACE FOR USE NOW: d59. TRANSITION FOR USE NOW: b2

158
I am in TRANSITION 52 now, which does SUPERSTATE.
I mark the OUTPUT place db9, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place db9

Object id: cb, Membership1: 0.50, Membership2: 1.00

CURRENT PLACES FOR USE: aR1 aR2 aR3 aR4 aR5 aR6 aR7 aR8 aR9 aR10
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 d58
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 d58
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 d58
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 d58
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 d58
PLACE FOR USE NOW: d68. TRANSITION FOR USE NOW: 52
Since transition 52 is consumed before, do not fire it again.
Since transition 52 is consumed before, do not fire it again.

I am in TRANSITION 8b now, which does CLIPPING.
I mark the OUTPUT place d/0, which is added to the tail of
CURRENT places list

TOKEN VALUE put in OUTPUT place d/0:

   Object id: cb, Membership1: 0.50

I am in TRANSITION 101 now, which does COMBINATION.
I mark the OUTPUT place d\106, which is added to the tail of
CURRENT places list

CLIPPED TOKEN VALUE put in OUTPUT place d\106:

   Object id: cb, Fuzzytermlist: cloudedstable, Clipping value: 0.50

I am in TRANSITION 102 now, which finds the overall FUZZY ACTION.
I mark the OUTPUT place d\107, which is added to the tail of
CURRENT places list

TOKEN VALUE put in OUTPUT place d\107:

   Object id: cb, Fuzzytermlist: cloudedstable, Overall Clipping value: 0.50
B.3 Forecasting the Weather Event - Active Rule Example (2)

INFERRENCE GROUP 1 FIRED FOR EXECUTION

CURRENT PLACES FOR USE: p92 p97 p98 p95
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: cb, Object attr value: 4.000000, Domain: fuzzycolor,
Fuzzytermist: 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: cb, Object attr value: cloudedstable, Domain: fuzzyweather,
Fuzzytermist: cloudedstable, Membership: 0.20

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 INFERRENCE GROUP 1, I am not using it.
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.
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: cb, Object attr value: 0.000000, Domain: fuzzywind,
Fuzzytermist: 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 p2b, which is added to the tail of CURRENT places list.
TOKEN VALUE put in OUTPUT place p2b
  Object id: cb, Object attr value: 22.500000, Domain: fuzzywinddr,
  Fuzzytermlist: south southwest, Membership: 0.00

CURRENT PLACES FOR USE: p99 p27 p24 p26 p2b p2b
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: cb, Object attr value: 22.500000, Domain: fuzzywinddr,
  Fuzzytermlist: north, Membership: 0.50

CURRENT PLACES FOR USE: p27 p24 p26 p2b p38
PLACE FOR USE NOW: p27, TRANSITION FOR USE NOW: 4b
I am in TRANSITION 4b now, which does COMPOSITION.
Since composition result is 0, rule rb is removed from the inference.

CURRENT PLACES FOR USE: p27 p24 p26 p2b 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: cb, Membership1: 0.25, Membership2: 0.00

CURRENT PLACES FOR USE: p24 p26 p2b p38 p37
PLACE FOR USE NOW: p24, TRANSITION FOR USE NOW: 4b
Since transition 4b is consumed before, do not fire it again.

CURRENT PLACES FOR USE: p24 p26 p2b 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 p2b 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: cb, Membership1: 0.2b

CURRENT_PLACES FOR USE: p26 p2b p38 p37 p112
PLACE FOR USE NOW: p26, TRANSITION FOR USE NOW: 4b
Since transition 4b is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: p2b p38 p37 p112
PLACE FOR USE NOW: p2b, TRANSITION FOR USE NOW: 4b
Since transition 4b is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: p38 p37 p112
PLACE FOR USE NOW: p38, TRANSITION FOR USE NOW: 4b
Since transition 4b 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: cb, Membership1: 0.2b

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: cb, Object attr value: 6.000000, Domain: Fuzzywind,
           Fuzzyterm1: calm breeze, Membership1: 1.00, Membership2: 0.2b
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: cb, Object attr value: -1.000000, Domain: fuzzytemp,
   Fuzzyterm list: cold, Membership1: 1.00, Membership2: 0.2c

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: cb, Object attr value: 185.000000, Domain: fuzzywinddr,
   Fuzzyterm list: south, Membership1: 0.67, Membership2: 0.2b

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: cb, Object attr value: 3.000000, Domain: fuzzyccolor,
   Fuzzyterm list: white, Membership1: 0.50, Membership2: 0.2b

CURRENT PLACES FOR USE: p46 p41 p42 p42
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: cb, Membership1: 0.25

CURRENT PLACES FOR USE: p41 p42 p43 p47

164
PLACE FOR USE NOW: p41, TRANSITION FOR USE NOW: 50

I am in TRANSITION 50 now, which does CUMPOSITION.

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: cb, Membership1: 0.50, Membership2: 0.25

CURRENT PLACES FOR USE: p42 p43 p47 p4c
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 p4c
PLACE FOR USE NOW: p47, TRANSITION FOR USE NOW: 94

I am in TRANSITION 94 now, which does CUMINATION.

I mark the OUTPUT place pl02, which is added to the tail of
CURRENT places list

CLIPPED TOKEN VALUE put in OUTPUT place pl02:

Object id: cb, Fuzzytermlist: fog, Clipping value: 0.25

CURRENT PLACES FOR USE: p40 pl02
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: cb, Membership1: 0.12

CURRENT PLACES FOR USE: pl02 p44
PLACE FOR USE NOW: pl02, 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 pl02
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: cb, Fuzzyterm list: 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: cb, Fuzzyterm list: fog, Overall Clipping value: 0.25
B.4 Querying the Temperature Change Forecast - Deductive Rule Example (2)

INFERENCES GNUP 2 FIRED FOR EXECUTION

CURRENT PLACES FOR USE: d85

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 d/2, which is added to the tail of
CURRENT places list

TOKEN VALUE put in OUTPUT place d/2

Object id: cb, Object attr value: cloudedstable, Domain: fuzzyweather,
Fuzzytermist: 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 d/3, which is added to the tail of
CURRENT places list

TOKEN VALUE put in OUTPUT place d/3

Object id: cb, Object attr value: 3.500000, Domain: fuzzypreschdir,
Fuzzytermist: 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 d/4, which is added to the tail of
CURRENT places list

TOKEN VALUE put in OUTPUT place d/4

Object id: cb, Object attr value: 22.500000, Domain: fuzzywinddr,
Fuzzytermist: 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: 1b

I am in TRANSITION 1b now, which does TRIGGERING condition.
I mark the OUTPUT place d/b, which is added to the tail of
CURRENT places list
TOKEN VALUE put in OUTPUT place d/5
   Object id: cb, Object attr value: 16b.000000, Domain: fuzzywinddr,
   Fuzzytermist: 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 d/8, which is added to the tail of
CURRENT places list

TOKEN VALUE put in OUTPUT place d/8
   Object id: cb, Object attr value: cloudestable, Domain: fuzzyweather,
   Fuzzytermist: cloudestable, Membership1: 1.00, Membership2: 1.00

CURRENT PLACES FOR USE: d89 d72 d73 d74 d75 d76 d77
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 d/9, which is added to the tail of
CURRENT places list

TOKEN VALUE put in OUTPUT place d/9
   Object id: cb, Object attr value: 3.500000, Domain: fuzzypreschdir,
   Fuzzytermist: increasing, Membership1: 1.00, Membership2: 1.00

CURRENT PLACES FOR USE: d89 d72 d73 d74 d75 d76 d77 d78
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 d/80, which is added to the tail of
CURRENT places list

TOKEN VALUE put in OUTPUT place d/80
   Object id: cb, Object attr value: 22.500000, Domain: fuzzywinddr,
   Fuzzytermist: north northwest northeast, Membership1: 0.50, Membership2: 1.00

CURRENT PLACES FOR USE: d89 d72 d73 d74 d75 d76 d77 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 d/81, which is added to the tail of
CURRENT places list

TOKEN VALUE put in OUTPUT place d/81

168
Object id: cb, Object attr value: 165.000000, Domain: fuzzywinddr,

Fuzzytermiilist: south southwest, Membership1: 0.67, Membership2: 1.00

CURRENT PLACES FOR USE: d89 d72 d73 d74 d75 d76 d77 d78 d79 d80
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
TUKEN VALUE put in OUTPUT place d84
Object id: cb, Object attr value: clouedstable, Domain: fuzzyweather,
Fuzzytermiilist: clear clearfew clouedinstable clouedstable, Membership1: 1.00,
Membership2: 1.00

CURRENT PLACES FOR USE: d89 d72 d73 d74 d75 d76 d77 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
TUKEN VALUE put in OUTPUT place d85
Object id: cb, Object attr value: 3.500000, Domain: fuzzypreschdir,
Fuzzytermiilist: nochange, Membership1: 0.00, Membership2: 1.00

CURRENT PLACES FOR USE: d89 a72 a73 a74 a75 a76 a77 a78 a79 a80 a81 a84 a86
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
TUKEN VALUE put in OUTPUT place d86
Object id: cb, Object attr value: 22.500000, Domain: fuzzywinddr, Fuzzytermiilist: west, Membership1: 0.00, Membership2: 1.00

CURRENT PLACES FOR USE: d89 d72 d73 d74 d75 d76 d77 d78 d79 d80 d81 d84 d86 d89
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: cb, Object attr value: 6.000000, Domain: fuzzywind,
Fuzzytermiist: breeze, Membership1: 1.00, Membership2: 1.00

CURRENT PLACES FOR USE: d72 d73 d74 d75 d76 d77 d78 d79 d80 d81 d82 d83 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 d76 d77 d78 d79 d80 d81 d82 d83 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 d76 d77 d78 d79 d80 d81 d82 d83 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 d76 d77 d78 d79 d80 d81 d82 d83 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: d76 d77 d78 d79 d80 d81 d82 d83 d84 d85 d86 d87
PLACE FOR USE NOW: d76. 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: cb, Membership1: 0.50, Membership2: 1.00

CURRENT PLACES FOR USE: d77 d78 d79 d80 d81 d82 d83 d84 d85 d86 d87 d77
PLACE FOR USE NOW: d77. TRANSITION FOR USE NOW: 54
Since transition 54 is consumed before, do not fire it again.

CURRENT PLACES FOR USE: d80 d81 d82 d83 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.

170
CURRENT_PLACES FOR USE: d83 d84 d85 d86 d87 d77
PLACE FOR USE NOW: d81. TRANSITION FOR USE NOW: b4
Since transition b4 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: b5
I am in TRANSITION b5 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: b5
Since transition b5 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d86 d87 d77
PLACE FOR USE NOW: d86. TRANSITION FOR USE NOW: b5
Since transition b5 is consumed before, do not fire it again.

CURRENT_PLACES FOR USE: d87 d77
PLACE FOR USE NOW: d77. TRANSITION FOR USE NOW: b7
I am in TRANSITION b7 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: cb, Membership: 0.56

CURRENT_PLACES FOR USE: d84
PLACE FOR USE NOW: d84. TRANSITION FOR USE NOW: b7
I am in TRANSITION b7 now, which does COMPOSITION.
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: cb, Fuzzytermlist: decrease, Clipping value: 0.56
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: cb, Fuzzyterm list: 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 that 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 the behavior. A class collects objects that share the same set of attributes and methods. Subclasses derive from 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 database 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, database 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 languages
- 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 objects in 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 envi-
nronment of a distributed database manager.

**Data Abstraction:** ITASCA supports data abstraction, i.e. the capability 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 oper-
ations on the data must use an external interface. Modules in an object database
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 manufacture unnecessary unique keys. ITASCA automatically supplies a unique identifier for abstract 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 an analogues 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 the 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 unavailable data has no impact on the work. Even changes to replicated objects (such as the schema) can occur because the ITASCA spools such changes 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 database exists on each server site. The user or application does not need to know the location of 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 the private database 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 database 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 failures 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 database transaction is atomic and durable.

**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 checkout 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 separate sessions. Separate 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 database 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 database 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 failure to a consistent database 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 writing 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 2nd 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


