



**A FORMAL APPROACH TO MODELLING AND
VERIFICATION OF CONTEXT-AWARE SYSTEMS**

A THESIS SUBMITTED TO THE UNIVERSITY OF NOTTINGHAM FOR

THE DEGREE OF

DOCTOR OF PHILOSOPHY

DECEMBER 2016

HAFIZ MAHFOOZ UL HAQUE

MSC. IT, MBA (SPECIALIZATION IN MIS)

SCHOOL OF COMPUTER SCIENCE

THE UNIVERSITY OF NOTTINGHAM

MALAYSIA CAMPUS

Abstract

The evolution of smart devices and software technologies has expanded the domain of computing from workplaces to other areas of our everyday life. This trend has been rapidly advancing towards ubiquitous computing environments, where smart devices play an important role in acting intelligently on behalf of the users. One of the subfields of the ubiquitous computing is context-aware systems. In context-aware systems research, ontology and agent-based technology have emerged as a new paradigm for conceptualizing, designing, and implementing sophisticated software systems. These systems exhibit complex adaptive behaviors, run in highly decentralized environment and can naturally be implemented as agent-based systems. Usually context-aware systems run on tiny resource-bounded devices including smart phones and sensor nodes and hence face various challenges. The lack of formal frameworks in existing research presents a clear challenge to model and verify such systems. This thesis addresses some of these issues by developing formal logical frameworks for modelling and verifying rule-based context-aware multi-agent systems. Two logical frameworks \mathcal{L}_{OCCS} and \mathcal{L}_{DROCS} have been developed by extending CTL^* with belief and communication modalities, which allow us to describe a set of rule-based context-aware reasoning agents with bound on time, memory and communication. The key idea underlying the logical approach of context-aware systems is to define a formal logic that axiomatizes the set of transition systems, and it is then used to state various qualitative and quantitative properties of the systems. The set of rules which are used to model a desired system is derived from OWL 2 RL ontologies. While \mathcal{L}_{OCCS} is based on monotonic reasoning where beliefs of an agent cannot be revised based on some contradictory evidence, the \mathcal{L}_{DROCS} logic handles inconsistent context information using non-monotonic reasoning. The modelling and verification of a healthcare case study is illustrated using Protégé IDE and Maude LTL model checker.

Acknowledgements

In the name of Allah, the Beneficent, the Merciful. All praise belongs to Allah, Lord of the worlds. And may blessings and peace be on Sayyidina Muhammad, the last of the Prophets, and his family and companions - all of them. I praise and Thanks Allah SubHanahu Wata'Ala for His countless and unbreakable showers of blessings upon me at every moment of time in my life. I got strengths, passion, inspiration and prudence just because of His very special kindness and blessings upon me which has enabled me to complete my PhD thesis today. Thanks Allah Almighty for everything.

My heartfelt gratitude goes to my respected supervisor Dr. Abdur Rakib for his excellent supervision, kind advices, suggestions, motivations and support. I am deeply indebted to him for his kind assistance during my PhD journey. Since the beginning of my PhD, he has always been taking interest in my research with full enthusiasms and dedications. I would like to thank Dr. Christopher Kroadnight for his assistance and suggestions. Many thanks goes to my colleagues in Computer Science research lab for their kind, love and support.

I am very much grateful to my honorable parents for their love, kind support, advices and great encouragement. Indeed my parents are unmatched, impeccable teachers and sagacious guardian and mentors who instilled many qualities in me which made my life fertile and enjoyable. I can never forget their worries and humble prayers. Truly, I have no words to thank them except prayers. May Allah Almighty bless them with uncountable showers of blessings here and hereafter. I would like to say a very special and heartfelt thanks to my wife for her support, patience, understanding and love during PhD journey. A special thanks goes to my sister and brothers for their prayers, moral and emotional support. Many thanks goes my respected elders and friends for their prayers and kinds. Finally, a lot of love and prayers for my beloved kids and nephews.

Contents

List of Figures	x
List of Tables	xii
1 Introduction	1
1.1 The notion of Context	2
1.2 Context-aware System	3
1.2.1 Context-aware Multi-agent Systems	6
1.2.2 Resources Constraints in Context-aware Systems	8
1.2.3 Resource-bounded Context-aware Agents	8
1.3 Motivation	9
1.4 Problem Statements	10
1.5 Methodology	11
1.6 Thesis Outline	13
1.7 Conclusion	14
2 Logical Formalisms for Multi-agent systems	15
2.1 Chapter Objectives	15
2.2 Introduction	15
2.3 Modal Logic	17
2.3.1 Syntax of Modal Logic	18
2.3.2 Semantics of Modal Logic	19

2.4	Temporal logic	20
2.4.1	Linear Temporal Logic (LTL)	21
2.4.1.1	Syntax	21
2.4.1.2	Semantics	21
2.4.2	Computational Tree Logic (CTL)	22
2.4.2.1	Syntax	23
2.4.2.2	Semantics	23
2.4.3	Full Computation Tree Logic (CTL*)	24
2.4.3.1	State Formula	24
2.4.3.2	Path Formula	25
2.4.4	Some Standard System Properties	26
2.4.5	Analysis of Temporal Logics	27
2.5	Model Checking	27
2.5.1	Maude LTL Model Checker	28
2.6	Conclusion	31
3	Formalisms for Context-aware MAS	32
3.1	Chapter Objectives	32
3.2	Reasoning Formalisms for the Semantic Web	32
3.3	Description Logic	34
3.3.1	DL Knowledge Base	36
3.4	The Semantic Web	38
3.4.1	Web Ontology Language (OWL)	41
3.4.2	OWL 2.0	42
3.4.2.1	Why to Choose OWL 2 RL	43
3.4.3	Why OWL is Not Enough?	44
3.4.4	Semantic Web Rule Language (SWRL)	44
3.4.4.1	DL Safe Rule in SWRL	47

3.5	Rule-based System	48
3.5.1	Components of Rule-based System	49
3.5.2	Multi-agent Rule-based System	50
3.6	Formal Approaches for Distributed MAS	51
3.6.1	Ontology-Based Context-aware Systems	52
3.7	Monotonic Vs Non-monotonic Reasoning	53
3.8	Defeasible Reasoning	55
3.8.1	Defeasible Logic	56
3.9	Defeasible Reasoning based Distributed Systems	58
3.9.1	Defeasible Reasoning based Frameworks for the Semantic Web	58
3.9.2	Integration of Description Logics with Defeasible Reasoning	60
3.9.3	Integrating Rules and Ontologies using DeLP	61
3.9.4	Defeasible Reasoning based Multi-context Systems	62
3.9.5	Discussion	63
3.10	Conclusion	63
4	The Logic $\mathcal{L}_{O CRS}$	65
4.1	Chapter Objectives	65
4.2	Motivation for the Logic $\mathcal{L}_{O CRS}$	65
4.3	Formal Approaches to Resource-bounded Multi-agent System	67
4.4	Description Logic Based Reasoning	68
4.4.1	Context Modelling	69
4.5	$\mathcal{L}_{O CRS}$ - A Logic for Context-aware Systems	72
4.5.1	The Language of $\mathcal{L}_{O CRS}$	74
4.5.2	Communication Bound	75
4.5.3	Memory Bound	75
4.5.4	Syntax	76
4.5.5	Semantics	77

4.5.6	Axiomatization	82
4.6	Soundness Proof	84
4.7	Completeness Proof	87
4.8	\mathcal{L}_{OCRS} Proofs of Correctness	90
4.8.1	The satisfiability Problem of $\mathbb{L}(n_M, n_C)$	92
4.8.1.1	The Canonical Model of $\mathbb{M}(n_M, n_C)$	92
4.8.2	Bisimulation	95
4.8.3	Soundness Proofs in the Canonical Model \mathbb{M}^c	98
4.8.4	Completeness Proofs in the Canonical Model \mathbb{M}^c	100
4.9	Encoding and Verification of \mathcal{L}_{OCRS} Model	102
4.10	Conclusion	103
5	The Logic \mathcal{L}_{DROCS}	104
5.1	Chapter Objectives	104
5.2	Motivation for the Logic \mathcal{L}_{DROCS}	104
5.3	Preliminaries	105
5.3.1	Non-monotonic Rule-based System	105
5.3.2	Defeasible Reasoning	106
5.3.3	Semantic Context Retrieval for \mathcal{L}_{DROCS}	108
5.3.4	Semantic Context Modelling	109
5.4	Context-aware Systems as Multi-agent Defeasible Reasoning Systems . .	110
5.5	The Logic \mathcal{L}_{DROCS}	111
5.5.1	Communication Bound	113
5.5.2	Memory Bound and Inconsistent Memory Manipulation	113
5.5.3	Syntax	114
5.5.4	Semantics	115
5.5.5	Axiomatization	120
5.6	Correctness Proof	122

5.6.1	Soundness Proof	122
5.6.2	Completeness Proof	123
5.7	A Simple Health-care Example	125
5.7.1	Verifying System Properties	128
5.8	Conclusion	131
6	Ontology-based System Modelling and Verification	132
6.1	Chapter Objectives	132
6.2	Motivation	132
6.3	Translation of an ontology into a set of Rules	133
6.3.1	Translating Ontology Axioms into DL Knowledge-base	133
6.3.2	Translating DL Knowledge-base into Defeasible Logic Program- ming (DeLP)	134
6.3.3	Translating Strict and Defeasible Terminologies to Horn-clause Rules	135
6.4	Onto-HCR Translator	137
6.4.1	OWL API	138
6.4.1.1	OWL API Design	138
6.4.1.2	Ontology Management	140
6.4.2	The Onto-HCR's Main Features	141
6.5	A Smart Environment: Case Study	144
6.5.1	Smart Environment Agents' Functions	145
6.6	Encoding and verification of $\mathcal{L}_{\mathcal{DROCS}}$ model	148
6.6.1	Maude Encoding	151
6.6.2	Specifying and Verifying the System	152
6.7	Conclusion	157
7	Conclusion and Future Work	158
7.1	Summary	158

7.2	Future Work	160
7.2.1	Extending Logic $\mathcal{L}_{\mathcal{DROCS}}$ Using Multi-Context System (MCS) . .	160
7.2.2	Contextualizing Ontologies	163
7.2.3	Distributed Semantic Knowledge Translator (<i>D-Onto-HCR</i>) . . .	164
7.2.4	Potential Application Framework using Context-aware Resource- bounded Devices	167
7.3	Conclusion	167
A	A set of rules for Smart Environment Case Study	192
A.1	Patient Care Agent	192
A.2	Blood Pressure Agent	196
A.3	Diabetes Tester	197
A.4	Body Temperature	198
A.5	Pulse Monitor	199
A.6	Ambulance Agent	200
A.7	Emergency Monitoring Agent	200
A.8	OnCall Agent	201
A.9	GPS Sensor	201
A.10	Telephone Agent	202
A.11	Caregiver Agent	202
A.12	Image Sensor	202
A.13	Motion Detector	203
A.14	Gas Detector	203
A.15	Glass Break Sensor	204
A.16	Smoke Sensor	204
A.17	Relative Sensor	204
A.18	Light Sensor	205
A.19	Occupancy sensor	205

A.20 Aircon Sensor	205
A.21 Temperature Level Sensor	206
B The <i>Onto-HCR</i> Translated Rules	208

List of Figures

3.1	OWL Family Tree	39
3.2	Structure of Rule-based System	50
4.1	A Fragment of the Epileptic Patients' Monitoring Ontology	69
4.2	Example SWRL Rules	70
4.3	Two Agent's Communication	73
4.4	Bisimilar Model Forth condition	96
5.1	Semantic Context Retrieval for \mathcal{L}_{DROCS}	108
5.2	A fragment of Home-care Patient's Monitoring System	126
5.3	Individualized Patient Ontology	126
5.4	Some SWRL Rules	127
6.1	OWL API Model	139
6.2	Onto-HCR Flow Chart	141
6.3	Main Menu	141
6.4	Some of the TBox Axioms	142
6.5	Some of the ABox Concepts Axioms	142
6.6	Some of the ABox Property Axioms	142
6.7	Some of the Horn-clause Rules	143
6.8	Context-aware Agents and their Possible Interactions	149
6.9	A Fragment of the Smart Environment Ontology	149
6.10	Idividualized Smart Environment Ontology	150

6.11	Some Rules of the Smart Environment Ontology	150
7.1	Multi-context awareness in the working memory of agent i	162
7.2	Class hierarchy of Smart Environment Ontologies	164
7.3	Distributed Semantic Knowledge Translation Process	165
7.4	<i>D-Onto-HCR</i> Output	166

List of Tables

3.1	Description Logic Constructors	38
3.2	Corresponding Terminologies of FOL, DL and OWL	40
3.3	OWL 2 RL Axioms	43
4.1	Horn-Clause Rules for the Epileptic Patients' Monitoring Context-aware System	70
5.1	Example Rules for a homecare patients' monitoring context-aware system	127
5.2	One Possible Run (Reasoning) of the System (Continued on Table 5.3) . .	129
5.3	One Possible Run (Reasoning) of the System	130
6.1	Translating OWL 2 RL Axioms into DL Knowledge-base	133
6.2	Mapping from DL Axioms to Strict and Defeasible Terminologies	136
6.3	Mapping from Strict and Defeasible Terminologies into Rules	137
6.4	Smart Environment Agent's Description	148
6.5	Experimental Results of the First System	154
6.6	Experimental results of the Second System	155
6.7	Experimental results of the Third System	156

List of Abbreviations

GPS	Global Positioning System
ABLS	Active Badge Location System
MAS	Multi-agent systems
PDA	Personal Digital Assistant
CoBrA	Context Broker Architecture
LTL	Linear Time Temporal Logic
OWL	Web Ontology Language
OWL API	OWL Application Programming Interface
OWL 2 RL	OWL 2 Rule language
SWRL	Semantic web rule language
CTL	Computational Tree Logic
CTL*	Full Computational Tree Logic
DL	Description Logic
DL-SHOIN	Language of description logic
DL-SROIQ	One of the most expressive description logic language
JADE	Java Agent Development Framework
HCR	Horn clause rules
FOL	First Order Logic

PTL	Propositional Temporal Logic
BNF	Backus Naur Form
RBS	Rule-based System
AI	Artificial Intelligence
ML	Modal Logic
KB	Knowledge Base
ALC	Attributive language with complements
ALB	Attributive language with Boolean algebras on concepts and roles
OWL 2 DL	OWL 2 description language
OWL 2 QL	OWL 2 query language
TBox	Terminological Box
ABox	Assertional Box
RBox	Relational Box
RIA	Role Inclusion Axioms
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
DAML	DARPA Agent Markup Language
OIL	Ontology Inference Layer
URI	Universal Resource Identifier
W3C	World Wide Web consortium
URL	Uniform Resource Locator
URN	Universal resource Name
DLP	Description Logic Program
NaF	Negation as Failure

DDL	Description Defeasible logic
DeLP	Defeasible Logic Programming
GCI	General Concept Inclusions
MCS	Maximal Consistent Set
CM	Canonical Model
XML	Extensible Markup Language
EBNF	Extended Backus-Naur Form
MP	Modus Ponens
Onto-HCR . . .	Tool Name
LP	Logic program
IRI	International Resource Identifier
Java IDE	Java Integrated Development Environment
JAR	Java Archive
$\mathcal{L}_{O\mathcal{C}RS}$	A logic for ontology driven context-aware resource-bounded systems
$\mathcal{L}_{\mathcal{D}ROCS}$	A logic for defeasible reasoning based ontology driven context-aware systems
O_{SPC}	Smart Patient Care Ontology
O_{SHO}	Smart Home Care Ontology
O_{SHP}	Smart Hospital Ontology
\mathcal{DL}_{SPC}	Smart Patient Care DL Knowledge base
\mathcal{DL}_{SHO}	Smart Home Care DL Knowledge base
\mathcal{DL}_{SHP}	Smart Hospital DL Knowledge base

Chapter 1

Introduction

Since the last decade, interests and demands for computing devices and smart applications have been expeditiously increasing. With their remarkable progression, these devices are becoming more sophisticated, optimized, complex, and smart. This trend is dynamically progressing towards ubiquitous computing. In this arena, numerous devices are impeccably integrated via portable or embedded devices by providing readily available services to facilitate users at anytime and anywhere. In ubiquitous computing environment, users exchange information using smart devices which made human life much easier, comfortable, and secure but device dependent. Everyday users spend much time and efforts on these devices to produce their desired objectives, however sometimes it becomes burdensome and monotonous. Context-aware computing is one of the most emerging and innovative paradigms to address these issues. It is a component of ubiquitous computing which aims to provide invisible computing environment to assist users in such a way that they can employ services whenever and wherever needed. More so, context-aware computing affords freedom from the bondage of traditional computing systems. In contrast with the traditional computing paradigm, context-aware computing environment is adaptive and highly dynamic in nature. These systems interact with human users; exhibit complex adaptive behaviours; and run on a highly decentralized environment. Context-aware systems facilitate computing devices to understand their context (situation or information)

and act on behalf of users. This significantly improves the user's productivity with the use of lesser effort. More so, services in ubiquitous environment is essentially required to become context-aware. This characteristic permits the adaptation in accordance with the rapid changes in the environment [Weiser, 1999, Zhang et al., 2005, LIRIS, 2010].

The aim of this research is to study logical frameworks for modelling, reasoning about, and verifying context-aware systems. The rest of this chapter is devoted for introducing the context-aware systems, motivation of developing such formalisms for context-aware systems, problem statements, methodology and research contributions. The structure of the thesis is outlined at the end of this chapter.

1.1 The notion of Context

In general, the term *context* has two dictionary meanings. Firstly, context means a word or a phrase or a text - used before or after a particular phrase and has fixed meanings. Secondly, context means a specific situation within which something happens or some events take place, and an action is taken accordingly. The first meaning is associated to linguistics whereas the second meaning is by some means closer to the definition of context in the world of computer science [Chen and Tolia, 2001]. Literature highlighted many definitions of context in different areas of computer science including ubiquitous computing [Chen and Tolia, 2001, Agre, 2001, Schilit et al., 1994], Sensor networks [Schmidt et al., 1999, Priyantha et al., 2000], Nomadic computing [Kindberg and Barton, 2001], Information retrieval [Castro and Muntz, 1999] and Artificial intelligence in different views [Schilit et al., 1994, Pascoe, 1998, Schmidt et al., 1999, Abowd et al., 1999, Strang et al., 2003]. Schilit and Theimer [Schilit and Theimer, 1994] define context as location, identities of people nearby, objects and changes made on these objects. In [Brown et al., 1997], context is defined as a location, identities, time, season, temperature etc. Ryan et al. [Ryan et al., 1999] define context in terms of user's location, identity, environment, and time. Context can also be represented in terms of atomic facts to express a certain

situation. In context-aware computing, the meaning of context does not change; however, their interpretation may vary or be modified [Chen and Tolia, 2001]. Dey et al. define context as “*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves*” [Abowd et al., 1999]. The notion behind is the awareness of context in context-aware systems. Hence, the context is relevant to a user and application. In due course, it reflects the relationship among them. In this thesis, we consider the above definition of context to grasp the intelligent behavior of context-aware systems. Context-aware systems acquire contextual information from various corresponding context-aware devices, evaluate it according to the specific criteria and then perform reasoning to achieve the desired goals.

1.2 Context-aware System

We, human beings, are blessed with the ability of context-awareness. With this, we often exploit contexts (ideas/messages) in our daily lives. These are effectively exchanged based on current situations without explicitly knowing the contextual information. Several situations reflect- the very fact that the core notion of the context directly affects the human intelligent behavior. With the rapid inventions in today’s modern world, computing devices are increasingly becoming more intelligent and smart. The evolution has been advancing towards the new generation of the systems under the name of context-awareness. Context sensing is one of the basic features of context awareness. Due to the intelligent behavior of today’s computing devices, context-awareness has simulated on these devices to reinforce their capability to acquire, exchange, process information, and adapt their behavior. For example, GPS (Global Positioning System) is a contextual sensing device. This calculates the longitude and latitude and then convey message representing the accurate location of the user with the annotated values or via a map. In-

dubitably, context-awareness has compelling uses for retrieving information from certain domains.

The research community has commonly agreed [Abowd et al., 1999] that the first research was done on context-aware computing in 1992 at Olivetti research Ltd, England [Want et al., 1992] in which Want et al. have developed Active Badge Location System (ABLS). This kind of badge is intended for their staff members to track their locations. On the other hand, these badges provide information about the current location of the cardholder to central hub thru signal transmission system using a network of sensors. Later in 1994, the concept of context-aware applications was first introduced by Schilit and Theimer [Schilit and Theimer, 1994] to adapt its behavior according to user's locations, nearby people and objects, as well as, changes made on these objects. Soon after, numerous attempts have been made on context-aware systems in different aspects. In context-aware system, the term *context-aware* has several meanings such as adaptive [Brown, 1996], reactive [Cooperstock et al., 1995], situated [Hull et al., 1997], responsive [Elrod et al., 1993], environment-based [Fickas et al., 1997] and context-sensitive [Rekimoto et al., 1998]. According to [Schilit and Theimer, 1994, Ryan et al., 1999], context-aware computing is defined as the ability of computing devices to detect and sense, interpret and respond to different aspects of users environment and devices themselves. In [Schilit and Theimer, 1994, Brown et al., 1997, Dey and Abowd, 2000, Dey et al., 1998], researchers have shown that context-aware applications exhibit dynamic changes. This ability enables the adaptation of behavior based on the situation of users and applications.

In recent years, rapid advances in the field of context-aware computing has significantly impacted on pervasive environment. In context-aware computing, context-awareness is the key requirement for developing context-aware systems. More so, context-aware systems can be easily trained to familiarize their operations in current context without explicit user interference. Also, context-aware applications are typically designed for the mobile users who use smart devices with embedded sensors. These embedded sensors could be

used to sense and acquire environmental contextual (low-level) data that have different interpretations according to nature of context-aware applications. For context adaptation, rather than providing only the contextual data from sensors, applications adapt their behavior based on the current contextual information.

In [Schilit et al., 1994], Schilit et al. have proposed two orthogonal dimensions for context-aware systems: manual and automatic. They have defined in both aspects whether the task is acquired and executed manually or automatically. In the first dimension, applications that obtain information manually based on available contexts for the user are known as proximate selection strategy. This technique is known as contextual command applications. This is owing to the fact that these applications normally execute commands manually based on available contexts for the users. The second dimension is the automatic contextual reconfiguration. These applications acquire information based on the available contexts for their users automatically. It is known as system level technique which produces the automatic binding to available resources based on current contexts. In this technique, whenever correct combination of contexts exists, this service automatically execute the context triggering actions which are based on simple if-then rules. Pascoe [Pascoe, 1998] has proposed the context adaptation feature which provides the ability to execute or update the service automatically based on current contexts. This feature is akin to context triggering actions.

Due to the rapid escalation of context-aware systems, the essential demand has increased for developing formal context models. This is to facilitate context representation in a variety of different heterogeneous systems [Esposito et al., 2008, Dey et al., 2001]. Contextual information usage is in the diversified fields with the embedded computing systems. Context sensing infrastructure, on the other hand, is intended to environment such as meeting rooms, class rooms, and home based applications. The most ambitious use of context-aware applications is to provide health care or general support services to elderly people within their home and facilitate them with the assisted living environment.

Various context-aware systems are developed to retain the independence of elderly people while safeguarding their emergency situations. Furthermore, these systems can quickly detect the emergency situations occurring with elderly people [Helal et al., 2003, Stanford, 2002, Intille et al., 2002, Bikakis et al., 2010, Leijdekkers and Gay, 2006]. Another importance of contextual information is its ability to automate the monitoring and control of environmental situations such as temperature setting and lighting [Lesser et al., 1999, Mozer, 1999] and remote operations of meeting rooms and class rooms [Bikakis and Antoniou, 2010].

The research presented, in this thesis, introduces a different vision of context-aware resource-bounded rule-based reasoning agents based on following automatic contextual reconfiguration technique in which context-aware agents perform rule-based reasoning for context triggering actions.

1.2.1 Context-aware Multi-agent Systems

The last two decades rapidly demonstrate the growth in the field of intelligent agents and agent-based reasoning. An intelligent agent is defined as a software system that perceives its environment and takes actions to perform specifically assigned tasks. Generally, multi-agents systems refer to software agents, in which, these agents are programed to solve specific problems. However, the agents in a multi-agent system could also be robots. Multi-agent systems are useful tools for modelling and solving large and complex problems. On the other hand, in ubiquitous computing, various heterogeneous computational entities (such as devices, services, applications and agents) perform certain tasks together to transform physical spaces into smart and interactive environment. Agent based approach is preferred among others because it performs two distinctive tasks. They sense the environment and reason on the current context in order to achieve the desired goal.

“We humans are inherently context-aware agents.”[Chen and Tolia, 2001]

Literature has revealed several context-aware systems from agents perspectives [Chen and Tolia, 2001, Kwon and Sadeh, 2004, Kwon et al., 2005, Chen et al., 2003, Fu and Fu, 2015]. In Context-aware computing, intelligent agents are designed in such a way that they anticipate the users' needs and act on their behalf. The behaviour of traditional agents is fixed based on certain set of environmental situations whereas the behavior of context-aware agents is dynamic which adapt changes based on the situation. More so, context-aware agents are software agents that are designed to model context-aware applications. These agents are designed and implemented in such a way that they have context-aware capabilities to function correctly and effectively based on sufficient knowledge and reasoning resources, which enable them to acquire information, reason about and adapt their behavior accordingly. For developing context-aware systems, the design and architecture of context-aware agents should be well-established to specify context-awareness features.

Context-aware multi-agent system paradigm is a suitable approach for the dynamic modular software design and for unpredictable environment. It provides well-established frameworks to specify, analyze, and implement complex software systems. Likewise, it enables them to act intelligently on behalf of the users. Context-aware agents possess decision-making capabilities that are realistic for highly dynamic environment [Koch and Rahwan, 2004]. Like traditional context-aware systems, context-aware multi-agent systems are designed in such a way that each agent in the system acquires certain contexts by performing inference or from other agents and then adapts its behaviour accordingly. Most of the traditional context-aware systems perform reasoning based on sensors' information. These systems have sophisticated sensing capabilities. In contrast, the design of context-aware multi-agent systems has the capability of accessing and reasoning ontological knowledge [Chen and Tolia, 2001].

1.2.2 Resources Constraints in Context-aware Systems

In the past few decades, various context-aware systems have been developed, however their functions remain primitive. This is because these systems are often more complex and costly due to their capability of context sensing and context reasoning [Strang et al., 2003, Want et al., 1992, Helal et al., 2003]. In context-aware pervasive computing environments, users usually have different mobile and smart devices to collect, process, and interpret information. Many context-aware systems often run on tiny resource-bounded devices and in highly dynamic environments. These context-aware devices such as smart phones, mobile phones, PDAs, GPS system, and different wireless sensor nodes, usually operate under strict resource constraints. Many challenges might arise when these context-aware devices communicate among themselves with the limited computational and communication resources. In this regard, we listed some constraints that often arise in context-aware systems.

- **Information Storage Constraints:** The memory space of storing and reasoning contextual information is often limited on most of the mobile devices. These devices are usually not privileged to store all contexts due to limited storage space.
- **Communication Constraints:** Context-aware devices often acquire contextual information from other smart devices. These devices communicate among themselves in a highly dynamic environment which causes quick reduction in battery energy level. However, most of the smart devices are not specifically designed to support this feature.
- **Time Constraints:** Mobile devices often have limited computational power. Its simultaneous execution of many programs make this process slower.

1.2.3 Resource-bounded Context-aware Agents

In the preceding section, we have discussed certain resource constraints of context-aware smart devices. Here, we list three basic resources that are required for agents in a multi-

agent context-aware systems to achieve goals.

- Time: It determines number of computational steps. It examines how many inference steps a system needs to perform, in parallel, to infer specific contextual information.
- Space: It determines the amount of memory required to store the initial and derived contexts. The size of the memory determines the number of cells in the memory to store arbitrary contexts. In order to solve a particular problem, it is important to know how much memory is required by an agent to store arbitrary contexts.
- Communication: It determines how many messages agents need to exchange in order to solve a given problem.

1.3 Motivation

Recent developments in the field of context-aware systems lead to a renewed interest in probing different approaches to modelling contextual information and reasoning context-aware systems [Bettini et al., 2010, Henricksen et al., 2002, Chen et al., 2003, Strang and Linnhoff-Popien, 2004]. The bulk of recent research in context-aware systems has focused towards various context modelling approaches such as logic-based modelling, ontology-based modelling, object-oriented modelling, key-value modelling, graphical modelling and Markup scheme modelling [Baldauf et al., 2007]. The ontology based context modelling is preferred among others due to its reasoning capability, simplicity, flexibility and extensibility, genericity, and expressiveness. Ontology based approach is considered as one of the most promising approaches in a ubiquitous computing environment. In the literature, some approaches have been proposed for ontology-driven context-aware systems, contextual reasoning, and context-aware learning service by using specific architecture and considering home health monitoring as an example system [Esposito et al., 2008, Hong and Cho, 2008]. In this thesis, we follow similar approaches for context

modelling proposed by [Esposito et al., 2008, Chen et al., 2003] .

The Context Broker Architecture *CoBrA* [Chen et al., 2003] allows distributed agents in a context-aware environment to have an access control to use their contextual information. The authors have described inference engine for reasoning with information expressed using the ontology. In [Ejigu et al., 2007], a context-aware system model facilitates the use of context-based reasoning by using ontology. This provides contexts, rules with their semantics. This model is designed for pervasive computing environment for re-usability and dynamicity due to resource limitation such as space and time which are crucial issues. Researchers have proposed various techniques for developing smart and reliable applications of context-aware systems [Zhang et al., 2005, Esposito et al., 2008, Nabih et al., 2011]. Agent based approach is one of them. In recent years, an ontology-based approach [Rakib and Faruqi, 2013] has been presented for the representation and verification of resource-bounded (time and communication) context-aware systems. This model is encoded with the Maude specification and properties of the system are verified using Maude LTL model checker.

To the best of our knowledge, none of the existing approaches provide a systematic logical framework for modelling and reasoning about context-aware systems with resource-bounds such as computational (time and memory) and communication resources. Thus, there is a need to develop a formal logical framework for context-aware resource-bounded (including memory, time, and communication) rule-based multi-agent system with their formal specification to model and reason smarter application domains.

1.4 Problem Statements

Considering today's modern world complex application domains, successful building of context-aware applications involves many challenging issues, including distributed problem solving, adaptability and autonomous behaviors. In fact, context-aware applications

are often based on tiny smart devices which operate under strict resource constraints. These systems become even more challenging in case of context-aware multi-agent systems when agents acquire context, reason about and exchange contextual information with limited computational and communication resources. Much effort has been made to provide solutions by integrating ontology with context-aware systems [Esposito et al., 2008, Chen et al., 2003, Rakib and Faruqi, 2013]. However, the focus on ontology driven context-aware logical frameworks with resource-bounds is still lacking. Based on the relevant literature discussed in Section 1.3, no systematic logical-framework has been proposed yet for resource-bounded context-aware systems.

This thesis aims to propose logical-frameworks for ontology driven resource-bounded context-aware multi-agent systems and verify interesting properties of the system. These frameworks use rule-based reasoning to reason contextual information using monotonic as well as non-monotonic reasoning based formalisms. The core emphasis is given in the distributed problem-solving for the systems of communicating context-aware rule-based agents and specify bounds (time, memory and communication) to achieve the desired goals. The key idea underlying the logical approach of context-aware systems is to define a formal logic that axiomatises the set of transition systems, and it is then used to state various *qualitative* and *quantitative* properties of the systems. For example, a *qualitative* property could be “*Can an agent have inconsistent beliefs (contradictory contexts in its working memory)*”, and *quantitative* properties could be “*an agent will always derive context φ in t time steps while exchanging fewer than n messages*” or “*every request of an agent i will be responded by agent j in t time steps*”, among others.

1.5 Methodology

An outline of the concrete methodology is given below:

1. We use ontology-based approach to represent contexts and rule-based reasoning to

infer implicit contexts from given a set of explicit contexts. We model a context-aware system as multi-agent system, where agents are reasoning agents and they reason over a knowledge-base using first order Horn-clause rules.

2. We develop a logical model, \mathcal{L}_{OCRS} , based on the previous work by [Alechina et al., 2006, Rakib et al., 2012, Alechina et al., 2009a, Alechina et al., 2009c] considering temporal epistemic and description logics. We interpret beliefs syntactically as formulas. The language of the logic contains a syntactic belief operator, temporal modalities to describe the transition system, and other modalities similar to those introduced in [Alechina et al., 2009a, Alechina et al., 2009c].
3. We improved \mathcal{L}_{OCRS} model using non-monotonic reasoning strategy. We present a logic \mathcal{L}_{DROCS} for context-aware non-monotonic reasoning agents. This work is based on [Gómez et al., 2007, Grosz et al., 2003] where context-aware agents use defeasible reasoning to reason with inconsistent information. We follow similar approach proposed by [Gómez et al., 2007, Grosz et al., 2003] while constructing a set of strict and defeasible rules from an ontology but our purpose and application of those rules are quite different. We use those rules to build a context-aware system as a multi-agent non-monotonic rule-based agents and use a distributed problem solving approach to see whether agents can infer certain contexts while they are resource-bounded.
4. We develop an OWL API based translation tool, *Onto-HCR*, which extracts OWL 2 RL and SWRL rules from an ontology and translate them into first order Horn-clause rules format.
5. We build a multi-agent rule-based context-aware system whose rules are derived from the ontology of a smart space scenario, which is adopted from [Bikakis et al., 2010, Leijdekkers and Gay, 2006, Nabih et al., 2011]. Then we show how we can encode and verify interesting properties of the systems using Model checking techniques, including non-conflicting contextual properties to see for example, when

there is an emergency situation for a patient then the system should not produce non-emergency situation at the same time.

1.6 Thesis Outline

This thesis is organized into seven chapters and the rest of the thesis is organized as follows:

Chapter 2 focuses on reviewing some basic logics with the intention of giving emphasis to the significance of how this literature can be compliant in putting forward and developing resource-bounded context-aware systems. We briefly summarize modal logic which is used to express many aspects of agents such as states, actions, and time. We then describe the linear and branching time temporal logics in order to understand model checking and verifying properties of the system. These logics are considered to be more appropriate for specifying, reasoning, and verifying multi-agent system. A brief overview of model checking and the foundation of Maude LTL model checker is given at the end of this chapter.

Chapter 3 presents the reasoning formalisms for the semantic web focusing on description logics with ontology languages and rule-based reasoning. We discuss the non-monotonic reasoning based formalisms, particularly, the defeasible reasoning which is used to reason inconsistent and incomplete information. We also have surveyed literature on monotonic as well as non-monotonic reasoning based logical formalisms including rule-based reasoning and the semantic web technologies. We also investigate the significance of rule-based approach for modelling ontology driven context-aware systems in this chapter.

Chapter 4 presents a formal logical framework for ontology-driven resource-bounded context-aware rule-based agents based on temporal epistemic description logic. We also

prove the correctness of $\mathcal{L}_{O\mathcal{C}\mathcal{R}\mathcal{S}}$ axiomatization such as soundness and completeness, validity and satisfiability and resource-bounded properties of the system.

Chapter 5 introduces the logic $\mathcal{L}_{D\mathcal{R}\mathcal{O}\mathcal{C}\mathcal{S}}$ which is the extended version of the logic discussed in the previous chapter. This work is based on non-monotonic reasoning formalism where agent's belief can be revised based on contrary evidence. We verify resource-bounded properties including non-conflicting contextual properties and illustrate the use of $\mathcal{L}_{D\mathcal{R}\mathcal{O}\mathcal{C}\mathcal{S}}$ framework using a simple health care case study.

Chapter 6 provides an appropriate translation process from ontology axioms to DL knowledge-base and then DL knowledge-base to defeasible logic programming. We present an OWL API based translator, *Onto-HCR*, for the logical frameworks and construct a comprehensive case study to model context-aware resource-bounded non-monotonic reasoning based system and verify its resource-bounded as well as non-monotonic properties. Finally, we demonstrate the scalability of the system by considering three facets of the system using model checking technique.

Chapter 7 provides the summary of the thesis and propose some of the possible future directions to extend this work as future work.

1.7 Conclusion

This chapter introduced the core notion of contexts and context-aware systems, and highlighted the state-of-the art of context-modeling and reasoning approaches. It also discussed motivation of the work presented in this thesis, followed by the problem statement and research objectives. Finally, outline of the rest of the thesis is described.

Chapter 2

Logical Formalisms for Multi-agent systems

2.1 Chapter Objectives

- To introduce the basic formalism of modal logics.
- To describe the linear and branching time temporal logics in order to understand model checking and verifying properties of the systems.
- To present a brief overview of model checking and Maude LTL model checking technique.

2.2 Introduction

In computer science, the aim of the logic is to develop a language that models a situation, in which, it can reason formally to realize the desired objective. A logical formalism has a language or a system of patterns continually behind a particular logic model with formal practices of reasoning and communication. Reasoning is a process of using existing knowledge or observations to make predictions or draw conclusions. Owing to basic techniques of logical formalisms, formal methods were born. Nowadays, formal methods are

extensively explored by researchers. These are used in the industry with their specification languages, theorem proving, and model checking [Huth and Ryan, 2004].

This chapter provides a brief survey of basic logical formalisms. The logics discussed in this chapter are very suitable to modelling and reasoning in real life application scenarios. From the time when these formalisms were born, they have still retained their grandeur due to versatile nature of modelling and reasoning multi-agent systems. These provide distinctive vantage for researchers with extensive interdisciplinary interests. Rather than just discussing only the influential language design of these logics with their powerful reasoning capabilities, these logics have incorporated various formalisms in other diversified fields [Alechina et al., 2009b, Huth and Ryan, 2004].

Before we introduce these formalisms, let us identify the essential drive of writing literature. We begin with the fundamental logics which provide very concise overview of propositional logic and first order logic in order to understand the modal logics, particularly comprehensive treatment of linear time and branching time temporal logics. These formalisms have greatly influenced the work presented in this thesis. Further, we have reviewed the literature on these logics for better understanding towards their interconnection with logical frameworks proposed in this thesis.

Propositional Logic is a formal language, which is often used in behavior analysis of computing systems. This is based on mathematical modelling that can be used to perform reasoning about the truthfulness or falsehood of logical expressions. The core idea of propositional logic is to develop a language to model the situation in a manner where reasoning can be performed formally to express properties of a system. It is based on propositions and propositional formulas, which are written in a propositional language. Its language includes the letters and logical connectives in terms of expressions or arguments, which is formally called propositional formulas. For example; the propositional clauses “It is hot” and “The sun is shining” can be represented using some propositional variables

p and q . Where the variable p can represent the first clause and q can be used to represent the second clause. In propositional logic, these clauses can be represented using logical connectives such as $p \wedge q$, among others, which means *It is hot* and *The sun is shining*. In general, propositional logic is a useful tool for modelling and reasoning in diverse application domains, especially in digital circuits. Most importantly, propositional logic is decidable. However, this logic may not be a suitable approach for modelling real life complex systems. In addition, propositional logic deals with simple declarative sentences. It has only Boolean values that may be either true or false and no quantifier variables are used in this logic, such as \exists, \forall [Huth and Ryan, 2004, Hedman, 2004].

First order logic (FOL) provides flexible and compact representation of knowledge. It consists of objects, properties, relations, and functions. Its domain of variables is very large and infinite. It can be distinguished from propositional logic in terms of its quantifiers. It is also known as symbolized reasoning since each sentence is splitted into subjects and predicates. FOL, on the other hand, is an extension of propositional logic. Additionally, it covers the predicates which may be either true or false. First order logic is more precise for representing predicate logic formulas and much more complex than propositional logic. In FOL, two sorts of things are considered to represent predicate logic formulas. Firstly, object represents the class with their individuals, for example, $A(x)$ and $B(x, y)$. Secondly, it shows the truth value. The downside of FOL is the fact that it is not decidable. However, the Description Logic is a subset of FOL which is decidable (discussed in Chapter 3). Thus, the first order logic formulas in the form of $A(x)$ and $B(x, y)$ can be directly translated into description logic concepts and roles respectively [Smullyan, 1995].

2.3 Modal Logic

At the outset, modal logic was introduced as a branch of logic in early 20th century. Since that time it was applied in different academic disciplines to analyze philosophical notions

and disputes. The core notion of modal logic is to study the truthfulness of a system. It is an extended version of propositional and first order predicate logic which includes modal operators in addition to other logical connectives to form complex formulas for modelling and reasoning. The basic language of modal logic is defined using a set of propositional variables with the modal operators. The propositional variables are usually denoted by letters p, q, r and so on. The modal operators, also known as modalities, are represented by \Box (necessity) and \Diamond (possibility). These operators are often read as: $\Box p$: *it is necessary that p* and $\Diamond p$: *it is possible that p* .

Applications of modal logic are characterized by many aspects of agent-based systems. This logic is considered to be one of the most widely used formal approaches to specify, reason about and verify multi-agent systems. It is also a compatible and versatile formalism for multi-agent systems to express various aspects of agents like beliefs, actions with their effects, and time. In a multi-agent system, an agent has a belief about certain facts that it believes to be true about the environment. The concept of belief was initially proposed by Hintikka (1962) [Hintikka, 1962] who describes the agent's belief as a set of possible worlds. There are two usual operators which describe the agent's belief [Huth and Ryan, 2004, Nga, 2011, Rakib, 2011].

1. The K_i operator arises from epistemic modality which is read as “*agent i knows that*”. Epistemic logic deals with the certainty of the formulas or sentences.
2. The B_i operator arises from doxastic logic which is read as “*agent i believes that*”. The truth of the formula p is expressed in terms of agent's belief as $B p$ in multi-agent system. More specifically, we say agent i believes a fact p , which is represented as: $B_i p$.

2.3.1 Syntax of Modal Logic

Formulas of basic modal logic are defined with traditional logical operators such as negation, disjunction and conjunction with modalities \Diamond and \Box . Let P be a finite set of propo-

sitional variables. The syntax is given as:

$$\phi ::= p \mid \neg \phi \mid \Diamond \phi \mid \Box \phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2$$

where $p \in P$. Other logical operators are given as: $\top \equiv \phi_1 \vee \neg \phi_2$ where \top represents true, $\perp \equiv \neg \top$ where \perp represents false, $\phi_1 \wedge \phi_2 \equiv \neg(\neg \phi_1 \vee \neg \phi_2)$, $\phi_1 \rightarrow \phi_2 \equiv \neg \phi_1 \vee \phi_2$, $\phi_1 \leftrightarrow \phi_2 \equiv (\phi_1 \rightarrow \phi_2) \wedge (\phi_2 \rightarrow \phi_1)$ and $\Diamond \phi \equiv \neg \Box \neg \phi$.

2.3.2 Semantics of Modal Logic

The semantics of Modal Logic are represented by Kripke model. Kripke model has the form $M = (S, R, V)$ where

- S is a non-empty set of states or possible worlds.
- The accessibility relation $R \subseteq S \times S$, which is a binary relation on S .
- $V: P \times R$ is a truth assignment function that may either be true or false.

The truth of the formula in a model $M = (S, R, V)$ and $s \in S$ is defined inductively as follows:

- $M, s \models p$ iff $V(p, s)$ is true.
- $M, s \models \neg \phi$ iff $M, s \not\models \phi$
- $M, s \models \phi_1 \vee \phi_2$ iff $M, s \models \phi_1$ or $M, s \models \phi_2$
- $M, s \models \phi_1 \rightarrow \phi_2$ iff $M, s \not\models \phi_1$ or $M, s \models \phi_2$
- $M, s \models \Box \phi$ iff for all $s' \in S$ such that $R(s, s')$, so $M, s' \models \phi$.

A formula ϕ is satisfiable iff there exists a model M and state s of M such that ϕ is at state s , $M, s \models \phi$. A formula ϕ is valid in a model M iff ϕ is true in any possible world s' of M . So we write it as: $M \models \phi$.

Here, we have briefly surveyed the core notion of Modal logics. We refer the interested readers to [Huth and Ryan, 2004, Benthem, 2010, Blackburn et al., 2002] for more detailed description of modal logics.

2.4 Temporal logic

In concurrent reactive systems, correctness not only validates the correct input and output of the computational system, but also objectively monitors the execution of the system. Temporal logic is explicitly developed to treat these aspects and to monitor infinite behavior of reactive systems. In this section, we describe temporal aspects of formal methods to model and specify multi-agent systems and verify their correctness properties. We elucidate how multi-agent systems are typically modelled and how reasoning using temporal logics is achieved. Temporal logics have played vital role in formal verification wherever needed to state the specification requirements for hardware and software systems. It represents the set of rules for reasoning in terms of time and the time domain is expressed in terms of state. A present time corresponds to current state and next moment of time corresponds to the immediate next state. Alternatively, system behavior is observed in terms of discrete time points such as $0, 1, 2, \dots, n$. Transition corresponds to the progressions from current time step to the next time step with specific action. Temporal logic is applicable due to its behavioral aspects of hardware and software in terms of time. Reasoning in the temporal logic is much easier with the translation into the predicate calculus because relationship among time is implicit [Baier et al., 2008].

Temporal logic has introduced some additional operators that represent the time variable and reflect their relationships. In essence, temporal logic is an extension of propositional logic. It is also known as propositional temporal logic (PTL). With the set of temporal operators, these enable the definition of formulas with the accessibility relation. When defining temporal properties, some temporal operators are needed to model the system in terms of time. These operators include F , G , X and U . The first three operators are

unary whereas the last one is binary [Wolper, 1983].

Literature has exposed several temporal logics including branching time and linear time temporal logics [Baier et al., 2008, Huth and Ryan, 2004, Pnueli, 1977]; each having their own temporal operators. In linear time temporal logic, there is a single successor state at each moment of time whereas in branching time temporal logic, time is split into different alternative paths. In the following section, we provide a brief overview of Linear temporal logic (LTL), Computational tree logic (CTL), and Full computational tree logic (CTL*).

2.4.1 Linear Temporal Logic (LTL)

Linear Temporal Logic has been proposed as a formal verification tool by Amir Pnueli [Pnueli, 1977]. LTL models time as a sequence of states and the future state is seen as a path in LTL. Accordingly, there are different future paths from where any path is taken as an actual path. There are many LTL model checking tools available, including [Eker et al., 2003, Barnat et al., 2006], which use LTL as a property specification language.

2.4.1.1 Syntax

The syntax of LTL is defined by considering a set of propositional variables (P), logical operators, and temporal operators. The set of LTL formulas is formally defined inductively over propositional variables as follows:

$$\phi ::= \top \mid p \mid (\neg\phi) \mid (\phi \vee \psi) \mid (\phi \wedge \psi) \mid (\phi \rightarrow \psi) \mid X\phi \mid F\phi \mid G\phi \mid \phi U \psi$$

The connectives F, X, R, G, U are known as temporal connectives where X (next) , F (eventually) , G (globally) and U (Until).

2.4.1.2 Semantics

We interpret temporal formulas in a linear model of time. The structure is formally represented by $M = (S, \rightarrow, L)$ where S is a set of states and L is a labeling function. The

semantics of a LTL formula is provided by the satisfaction relation (\models). $M, \pi \models \phi$ means the path π in M satisfies the formula ϕ . π expresses the paths as $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \dots \rightarrow s_n$ and $\pi^i = s_i \rightarrow \dots \rightarrow s_n$ is a sub path of π starting at s_i . The semantics is defined by induction on the structure of ϕ , is given as:

- $(M, \pi) \models \top$
- $(M, \pi) \models p$ iff $p \in L(s_0)$
- $(M, \pi) \models \neg\phi$ iff $(M, \pi) \not\models \phi$
- $(M, \pi) \models \phi \vee \psi$ iff $(M, \pi) \models \phi$ or $(M, \pi) \models \psi$
- $(M, \pi) \models \phi \wedge \psi$ iff $(M, \pi) \models \phi$ and $(M, \pi) \models \psi$
- $(M, \pi) \models \phi \rightarrow \psi$ iff $(M, \pi) \not\models \phi$ or $(M, \pi) \models \psi$
- $(M, \pi) \models X\psi$ iff $(M, \pi^1) \models \psi$ (Next step must be true)
- $(M, \pi) \models \phi U \psi$ iff there exist $i \geq 0$ such that $(M, \pi^i) \models \psi$ and for all $j < i$, $(M, \pi^j) \models \phi$

Other additional temporal operators can be expressed in terms of the above formulas [Huth and Ryan, 2004, Artale, a, Artale, b].

2.4.2 Computational Tree Logic (CTL)

CTL is a branching time temporal logic. Like LTL, it is also used in formal methods to verify correctness properties of a system by model checking tools. Each state may have an accessibility relation with the immediate next state by a transition which reflects the system behavior in terms of time. CTL is more expressive than LTL and permits quantification over path [Huth and Ryan, 2004].

2.4.2.1 Syntax

The syntax of CTL is defined by considering a set of propositional variables (P), logical operators, and temporal operators. The set of CTL formulas is formally defined inductively over propositional variables as follows:

$$\phi ::= \top \mid p \mid (\neg\phi) \mid (\phi_1 \wedge \phi_2) \mid (\phi_1 \vee \phi_2) \mid (\phi_1 \rightarrow \phi_2) \mid (\phi_1 \leftrightarrow \phi_2) \mid EX\phi \mid EG\phi \mid EF\phi$$

where $p \in P$. In CTL, each temporal connective is a pair of symbols. The first symbol in a pair is one of the E and A . E means along at least one path, and A means along all paths. The second symbol in the pair is X (next), F (eventually), G (globally) and U (Until). The symbols X , F , G and U cannot appear without an E or an A . These are used with logical operators in the same way as with other logics, these operators are: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

2.4.2.2 Semantics

CTL formulas and their interpretations are represented over transition system. A transition system is a triple $M = (S, \rightarrow, L)$ where S is the set of states, $\rightarrow \subseteq S \times S$ is a transition relation and L is labeling function. The relation of semantic entailment $(M, s \models \phi)$, for a given state $s \in S$ and a formula ϕ , is defined by structural induction on ϕ .

1. $(M, s) \models \top$
2. $(M, s) \models p$ iff $p \in L(s)$
3. $(M, s) \models \neg\phi$ iff $(M, s) \not\models \phi$
4. $(M, s) \models (\phi_1 \wedge \phi_2)$ iff $(M, s) \models \phi_1$ and $(M, s) \models \phi_2$
5. $(M, s) \models (\phi_1 \vee \phi_2)$ iff $(M, s) \models \phi_1$ or $(M, s) \models \phi_2$
6. $(M, s) \models (\phi_1 \rightarrow \phi_2)$ iff $(M, s) \not\models \phi_1$ or $(M, s) \models \phi_2$
7. $(M, s) \models EX\phi$ iff for some s_1 , such that $s \rightarrow s_1$, we have $(M, s_1) \models \phi$. Thus ϕ is in some next states.

8. $(M, s) \models EG\phi$ iff there is a path $s_1 \rightarrow s_2 \rightarrow \dots$ where $s_1 = s$ and for all s_i along the path such that $(M, s_i) \models \phi$.
9. $(M, s) \models EF\phi$ iff there is a path $s_1 \rightarrow s_2 \rightarrow \dots$ where $s_1 = s$ and for some s_i along the path such that $(M, s_i) \models \phi$.

2.4.3 Full Computation Tree Logic (CTL*)

CTL* is considered as a superset of CTL and LTL. CTL* is a powerful temporal logic which combines both the linear time and branching time temporal operators including state formulas with path quantifiers. It is much more expressive than CTL and LTL and freely combines the path quantifiers with temporal operators. CTL* temporal operators (F, X, G, U) are associated with unique path quantifier (E, A) [Huth and Ryan, 2004].

2.4.3.1 State Formula

CTL* have the same operators used in CTL. CTL* syntax is defined using propositional variables (P), logical operators, and temporal operators. The formulas of CTL* is formally defined as follows:

$$\phi ::= \top \mid p \mid (\neg\phi) \mid (\phi_1 \wedge \phi_2) \mid A(\phi) \mid E(\phi) \mid AF\phi \mid EF\phi \mid AG\phi \mid EG\phi \mid A(\phi_1 U \phi_2) \mid E(\phi_1 U \phi_2)$$

where $p \in P$ and ϕ is any path formula.

Kripke structure is used to define the semantics of CTL*. Two types of formulas used in CTL* are state formulas and path formulas. State formulas are interpreted with respect to states whereas path formulas are interpreted over paths.

A transition system is a triple $M = (S, \rightarrow, L)$ where S is the set of states, \rightarrow is a transition relation and L is labeling function. If a state s satisfies a formula ϕ , then it can be represented in a model at a state s , $(M, s \models \phi)$, given as:

- $(M, s) \models \top \wedge (M, s) \not\models \perp$
- $(M, s) \models p$ iff $p \in L(s)$
- $(M, s) \models \neg\phi$ iff $(M, s) \not\models \phi$
- $(M, s) \models \phi_1 \wedge \phi_2$ iff $(M, s) \models \phi_1$ and $(M, s) \models \phi_2$
- $(M, s) \models \phi_1 \vee \phi_2$ iff $(M, s) \models \phi_1$ or $(M, s) \models \phi_2$
- $(M, s) \models \phi_1 \rightarrow \phi_2$ iff $(M, s) \not\models \phi_1$ or $(M, s) \models \phi_2$
- $(M, s) \models \phi_1 \leftrightarrow \phi_2$ iff $\{(M, s) \models \phi_1 \text{ and } (M, s) \models \phi_2\} \text{ or } \{\neg(M, s) \models \phi_1 \text{ and } \neg(M, s) \models \phi_2\}$
- $(M, s) \models A\phi$ iff ϕ is a path formula, and for all paths π starting from s s.t. $(M, \pi) \models \phi$.
- $(M, s) \models E\phi$ iff ϕ is a path formula, and there is a path π starting in s s.t. $(M, \pi) \models \phi$

2.4.3.2 Path Formula

$$\phi ::= p \mid (\neg\phi) \mid (\phi_1 \wedge \phi_2) \mid (\phi_1 \vee \phi_2) \mid (\phi_1 \rightarrow \phi_2) \mid (\phi_1 \leftrightarrow \phi_2) \mid (\phi_1 U \phi_2) \mid G\phi \mid F\phi \mid X\phi$$

The semantics of CTL* Path formulas are defined as follows:

- $(M, \pi) \models p$ iff $(M, s_0) \models p$
- $(M, \pi) \models \neg\phi$ iff $(M, \pi) \not\models \phi$
- $(M, \pi) \models \phi_1 \wedge \phi_2$ iff $(M, \pi) \models \phi_1$ and $(M, \pi) \models \phi_2$
- $(M, \pi) \models \phi_1 \vee \phi_2$ iff $(M, \pi) \models \phi_1$ or $(M, \pi) \models \phi_2$
- $(M, \pi) \models \phi_1 \rightarrow \phi_2$ iff $(M, \pi) \not\models \phi_1$ or $(M, \pi) \models \phi_2$

- $(M, \pi) \models \phi_1 \Leftrightarrow \phi_2$ iff $\{(M, \pi \models \phi_1) \wedge (M, \pi \models \phi_2)\} \vee \{\neg(M, \pi \models \phi_1) \wedge \neg(M, \pi \models \phi_2)\}$
- $(M, \pi) \models X \phi$ iff $(M, \pi^1) \models \phi$
- $(M, \pi) \models F \phi$ iff there exist $n \geq 0$ such that $(M, \pi^n) \models \phi$
- $(M, \pi) \models G \phi$ iff for all $n \geq 0$ such that $(M, \pi^n) \models \phi$
- $(M, \pi) \models \phi_1 U \phi_2$ iff there exist $n \geq 0$ such that $(M, \pi^n) \models \phi_2$ and for all $0 \leq k < n$, so, $(M, \pi^k) \models \phi_1$

2.4.4 Some Standard System Properties

There are various system properties that can be expressed using temporal logics. Some of the generic properties are provided in this section [Alpern and Schneider, 1985, Rakib, 2011].

- *Safety property* describes “bad thing” that does not happen during the execution. According to safety property: “something bad will never happen”. It can be expressed as non-reachability of a state satisfying φ , e.g.; the property $AG \neg \varphi$
- *Liveness property* specifies about a “good thing” happens during the execution or we can say that “something good will eventually happen” which means that eventually some formula φ holds for finite number of steps. e.g.; $EF \varphi$
- *Guarantee of Service* states that if one process sends a request, it must have to be responded by other process. For example, if a request having a formula (e.g.; φ). Then, it must eventually be responded by a formula (e.g.; ψ). This can be expressed in temporal notation as: $AG(\phi \rightarrow AF \psi)$.
- *Mutually exclusion* occur when two processes execute concurrently at the same critical section. Then, it is prescribed as “bad thing”.

- *Precedence* is defined by *until* operation which states that a formula φ satisfies all preceding states until ψ will eventually happen.

2.4.5 Analysis of Temporal Logics

Three temporal logics discussed above are different in terms of expressive powers, quantifiers, structures and behavior of the systems. Linear Time Temporal Logic (LTL) is more straight forward to use and have the individual paths (Path formulas). Computational Tree Logic (CTL) has the ability to quantify paths. Its formulas like $AF p$ seems hard to understand. CTL restricts in two ways as compared to CTL*. It disallows boolean combination of path formulas and nesting of path modalities such as F, X, G . CTL* is more expressive than CTL and LTL but computationally it is much more expensive. CTL* is very useful for developing and checking correctness of complex reactive system.

2.5 Model Checking

“Model checking is an automated technique that, given a finite-state model of a system and a formal property, systematically checks whether this property holds for (a given state in) that model”[Baier et al., 2008]. Model checking is a technique to verify the correctness properties of a finite state system. In model checking, system checks whether the model meets the given specification or not. There are two types of specification: system specification and property specification. System specification analyzes the concurrent transition system to be formalized. In property specification, properties of the system are model checked [Baier et al., 2008, Eker et al., 2003]. Model checking is based on temporal logics in which properties are typically written using the temporal logic formulas. Literature has revealed considerable amount of work on model checking multi-agent systems considering temporal aspects of the system [Bordini et al., 2003, Wooldridge et al., 2002, Van der Hoek and Wooldridge, 2002] in general, and with resource-bounded in particular [Albore et al., 2006, Alechina et al., 2007, Rakib et al., 2012].

In the literature, a significant number of model checking tools are available to model, specify and verify the correctness properties of the systems. Some of the model checking tools that are most commonly used for verification of multi-agent systems such as MCK [Gammie and van der Meyden, 2004], MCMAS [Lomuscio et al., 2009], MOCHA [Alur et al., 1998], SPIN [Holzmann, 1997] and Maude LTL Model Checker [Eker et al., 2003] etc. Among others, we choose Maude LTL Model Checker due to its modular structuring mechanism. In addition, Maude performs model checking on finite state model and has the ability to check the systems whose states involves algebraic data types. In this section, we briefly introduce the basic foundation of Maude LTL model checker. However, in Chapter 6, we describe how to perform automated analysis of context-aware multi-agent system (a $\mathcal{L}_{\mathcal{DROCS}}$ model) and verify interesting properties of the system using the Maude LTL model checker.

2.5.1 Maude LTL Model Checker

This section presents the core foundation of Maude and Maude LTL model checking [Clavel et al., 2007, Eker et al., 2003]. Maude supports rewriting logic computation and deals with the concurrent computations. In Maude, systems are developed (modelled) using equational and rewriting logic. Maude is not only a modelling tool but also a high-performance programming language that models the system to specify the actions within the systems. It is simple, concrete and a powerful language that has the ability to present the system behavior. It supports both the equational and re-writing specification language and programming for multi-domain applications.

Maude has a modular structure to represent the basic units of specification as well as programming. In Maude, a module can be defined as a collection of sorts and a set of operations on these sorts. Sort is a category for values, which is declared in the module with the keyword *sort* followed by sort name and period at the end. Multiple sorts can be defined using the keyword *sorts*. Subsort is a subcategory of sort. It is like a subset

relation with sort which is used to define more specified group that belongs to the sort. It is declared using the keyword *subsort*.

A term can be a constant, variable, or an application of the operator that list the arguments of the term. However, a ground term is quite different from a normal term for it does not contain variables but only has constants and operators. Maude variable is used to define dynamic value for the sort. In Maude, a variable cannot be a constant. Maude variables are declared using the keywords *var* or *vars*. *var* is used to define a single variable whereas *vars* is used for multiple variables. Variable are the placeholders for terms.

In Maude, an operator is considered as a function or constructor to process data. Maude operators understand both the prefix or mixfix notations. Prefix operations are widely used by many programming languages whereas mixfix operations used in Maude. Maude operation is declared using keyword *op* followed by name, followed by colon(:), and followed by list of sorts, followed by an arrow (->), followed by sort for results and a period at the end.

In Maude, equations are used to describe the static part of the system. The dynamic part of the system, i.e., rewriting rule is one of the most powerful feature of Maude for the concurrent transitions that take place in the structure of the to/from states. Two types of equations are used in Maude: unconditional equations and conditional equations. Unconditional equations can be represented as:

$$eq \ [<LabelName>]:<Term-1>=<Term-2> \ [<OptionalStatementAttributes>] .$$

Conditional equation has some conditions with the equations, which is of the form:

$$ceq \ [<LabelName>]:<Term-1> = <Term-2> \\ if \ <EqCond-1> \wedge \dots \wedge \ <EqCond-k> \ [<OptionalStatementAttributes>] .$$

Unconditional rules can be defined using the following syntax:

$$rl \ [<LabelName>]: \ <Term-1> \Rightarrow \ <Term-2> .$$

Conditional rules can be defined using the following syntax:

```
crl [<LabelName>]:<Term-1> => <Term-2>
if <RuleCond-1> ∧ ... ∧ <RuleCond-k> .
```

In Maude, modules can be classified as functional modules, system modules and object-oriented modules. In this thesis, we discuss functional module and system module. These modules are declared with the key terms as follows:

```
fmod <ModuleName> is
```

```
...
```

```
endfm
```

```
mod <ModuleName> is
```

```
...
```

```
endm
```

Functional module begins with the keyword *fmod* and ends with keywords *endfm*. However, system module's structure is different, it starts with a *mod* keyword and end with *endm*. The *<ModuleName>* represents the name of the module and '...' represents the body of the module. More specifically, it represents all the declarations and statements within its scope. Moreover, the body of a functional module defines operations and data types through the use of the equational theory. Whereas the body of a system module's task is to specify a rewrite theory that contains the equational theory and rewrite rules. Maude requires a system specification and a property specification to verify properties using model checking tools. In Maude, system specification is defined by rewrite theory and property specification is specified by LTL formulas. In Maude, a module can be imported into another module based certain circumstances. For example, if a module *A* imports another module *B*. In that case, the module *A* does not need to re-define its declarations.

In Maude, rewrite theory represents the system dynamics which has concurrent transitions. In Maude rewriting system, the system explores properties non-deterministically using concurrent transitions of the distributed systems. Maude chooses arbitrarily the suitable rewrite rule that would be applied from left to right of the equation. The data types can be defined through its algebraic equations and the dynamic behavior of the system is specified by rewritable rules. A rewrite theory is more likely to become non-deterministic and could exhibit different behaviors.

Maude LTL model checker has the ability to check the systems whose states involve arbitrary algebraic data types. Due to that reason, we have chosen Maude LTL Model checker. However, there is one assumption - the set of states accessible from a given initial state is finite.

2.6 Conclusion

In this chapter, we have mainly focused on reviewing some basic logics with the intention of giving emphasis to the significance of how this literature can be compliant in putting forward and developing resource-bounded context-aware systems. Propositional logic was developed as a formal language, which has the capability to manipulate and reason using inference rules. However, it is often not suitable for modelling real life example systems. This logic does not have quantification. Thus, it is very difficult to express the large domains concisely. To begin with, first order logic has quantifiers which are naturally more expressive than propositional logic. Secondly, modal logic is used to express many aspects of agents such as mental states, actions, and time. This logic is considered to be more appropriate for specifying, reasoning, and verifying multi-agent system. Temporal logic is used for reasoning about concurrent programs of multi-agent systems and is very useful for formal verification of hardware and software requirements. At the end of this chapter, we have presented the basic foundation of Maude LTL model checker, which will be used to verify formally properties of context-aware systems discussed in this thesis.

Chapter 3

Formalisms for Context-aware MAS

3.1 Chapter Objectives

- To present the basic formalisms of description logics (DLs) and explain why DL-based ontologies are important in the semantic web.
- To introduce the web ontology languages (OWL), with their data type formalisms, and Semantic web rule language (SWRL).
- To investigate the significance of rule-based approach for modelling ontology driven context-aware systems.
- To discuss the non-monotonic reasoning based formalisms, particularly, the defeasible reasoning.

3.2 Reasoning Formalisms for the Semantic Web

In this chapter, we introduce the core notions and notations of the reasoning formalisms employed in this thesis. Reasoning is denoted as the process of thinking in a logical way in order to draw inference or conclusion. When reasoning is performed, it draws inference, which may be either monotonic or non-monotonic. More so, this chapter provides a

comprehensive review of both kinds of reasoning formalisms, focusing on two important aspects of the research problems. The first aspect introduces the reasoning formalisms for the semantic web, in this part, we discuss two different but contingent areas of knowledge representation formalisms. First area focuses on description logics (DLs), which are widely used and best suited for ontology languages and have variety of applications. The core inspiration of DLs is to develop a well-suited integrated computational services to model the domain using ontology. Description logic based ontology has a significant impact in the semantic web owing its expressivity and powerful DL reasoning.

The second area is concerned with web ontology languages (OWLs), which are the formal representation of a conceptualization for a specific domain that defines certain concepts/classes and their relationship. There are two versions of ontology languages known as OWL (OWL 1) and OWL 2, each having their own sub-languages. In Section 3.4.4, we discuss semantic web rule language (SWRL) which was proposed to overcome the limitations of OWL 2 by defining the semantic relations among individuals and it enhances the expressive power of OWL. In this thesis, we explicitly address the semantic context modelling approach by combining of OWL 2 RL with SWRL.

In Section 3.5, we present rule-based reasoning formalisms following semantic web technologies and then describe the formal approaches for distributed multi-agent system in the proceeding section. In Section 3.7, we provide a brief comparison between monotonic and non-monotonic reasoning techniques. Thereafter, we proceed to defeasible reasoning which is one of the most successful area in non-monotonic reasoning. Section 3.9 briefly surveyed some ontology driven defeasible reasoning based formalisms which include rule-based reasoning techniques for modelling and reasoning. Finally, we provide the conclusion in the last section.

3.3 Description Logic

Description Logics (DLs) are a family of formal knowledge representation languages which is specifically designed to express terminological knowledge of a particular domain in a well-structured and organized manner and are best suited for expressing and reasoning with regards to knowledge in a given application domain. The advances in the development of description logic languages has gained significant popularity due to their expressivity and computational complexity in knowledge representation formalisms such as the semantic web, ontology-driven data access, biomedical informatics, etc [Tsarkov and Horrocks, 2006, Hustadt et al., 2004, Botoeva, 2014]. DLs model individuals, atomic concepts and roles. It also defines their relationships. It is used for formal reasoning on atomic concepts of a particular application domain. In the proceeding Subsection 3.3.1, we discuss the DL knowledge which is the basic building block of description languages.

Description logic is a fragment of First Order Logic which is exclusively designed to formalize logic-based systems with semantic networks. In the past years, much more information was made available on description logics. DLs, on the other hand, have been applied in different areas of computer science like in ontology modelling for semantic web, knowledge representation and data integration [Baader, 2003, Grosz et al., 2003]. Accordingly, first order logic is the best suited framework for analyzing first order definable logic; hence, the mapping between description logic and FOL is straightforward. Then, the mapping from DLs and modal logic to first order logic joined them using powerful technique proposed in the field of automated reasoning. The DL language is considered to be a sub-language of the universal terminological logic. The universal terminological logic has been studied in 1987 by Patel-Schneider [Patel-Schneider, 1987]. There are many supersets of operators available in most description logic languages, which can be found in the literature [Hustadt et al., 2004].

Literature has revealed variety of sub logics of DLs [Baader et al., 2003, Baader and Nutt, 2003, Calvanese and De Giacomo, 2003]. For instance, Attributive language with complements(ALC) [Schmidt-Schauß and Smolka, 1991, Rudolph, 2011] is one of the initial versions of DL. This is limited to top and bottom concept, concept complement, concept intersection and existential quantifiers. However, the core modelling features of ALC have been enriched by its expressive formalism to specify and query knowledge. The features such as RBox axioms, the universal role, cardinality constraints, role inverse, self-concepts and nominal-concepts are not supported by ALC. Later in 2000, Hustadt et al. [Hustadt and Schmidt, 2000] have focused on a decidable description logic ALB which is an abbreviation of attributive language with Boolean algebras on concepts and roles. It extends ALC with the top role, role union, role intersection, role complement, role inverse, range restriction, and domain restriction. ALCH or SH is the extension of [Schmidt-Schauß and Smolka, 1991] which additionally allows simple role inclusion such as role chain axioms ($R \sqsubseteq S$).

Horrocks et al. [Horrocks et al., 2006] have proposed description logic SROIQ based on ontology language OWL 2 DL and is considered as one of the most expressive formalisms where inferencing is decidable. It has three primary elements termed as concept names, role names, and individual names. The basic building blocks of DL SROIQ are RBox (shows the inter-dependencies between the roles), TBox (introduces terminology or vocabulary of an application domain) and ABox (contains information that applies to individuals). In DL SROIQ, concepts $C, D \in \mathcal{C}$ are said to be equivalent, represented as $C \equiv D$, if both concepts C and D are having the same extension for the interpretation I , for example; $C^I \equiv D^I$. The set of all concepts is represented by \mathcal{C} in the knowledge-base (KB). Concept equivalence is expressed in the form of axiom entailment, for example, $C \equiv D$ means a knowledge base has both similar axioms $D \sqsubseteq C$ and $C \sqsubseteq D$. i.e $KB \models C \sqsubseteq D, KB \models D \sqsubseteq C$. Concept equivalence $C \equiv D$ can be expressed as TBox axiom in the knowledge base. In a similar fashion, two knowledge bases are equivalent if the

interpretation of first knowledge base exactly matches with the second knowledge base axioms.

Description logics express formal logic based semantics. Semantics of concept identifies Description Logics, including DL SROIQ, as DL is semantically based on first order predicate logic. More specifically, some DL interpretation has the same structure as FOL interpretation. But technically speaking, DL syntactic structure is different from FOL and most DL axioms are not FOL formulas. DL atomic concepts and roles can be mapped to unary and binary predicates respectively. Any predicate logic formula in FOL $C(x)$ can be mapped by DL concept C where C is the class name and x is a variable. In DLs, concepts are the set of individuals presented by letters C, D . Roles are represented by binary relation between individuals R and a, b are individuals for the concepts or roles. Concept assertion and role assertions are represented by $C(a)$ and $R(a, b)$ respectively. The DL axioms can be translated to equivalent FOL formulas. [Baader, 2003, Horrocks et al., 2006]

- An atomic concept C can be translated to $C(x)$
- An atomic role R can be translated to $R(a, b)$
- Constructor intersection is translated to logical conjunctions
- Union is translated to disjunction
- Negation is translated to Negation

3.3.1 DL Knowledge Base

DL knowledge base has a finite set of terminological and assertional sentences having three essential components:

1. **TBox (Terminological Box)** presents the terminology of an application domain. TBox defines the concepts, it specifies concept hierarchies which shows how atomic

concepts and atomic roles are interrelated. TBox outlines the controlled vocabulary in the form of classes and properties. The basic form of declaration in TBox concepts and roles are given as follows:

- Concept Inclusion: $C \sqsubseteq D$
- Role Inclusion: $(R \sqsubseteq S)$.
- Concept Equivalence: $C \equiv D$
- Role Equivalence: $(R \equiv S)$.

2. **ABox (Assertional Box)** is an assertional component that describes facts associated with concepts and roles in the knowledge base. ABox contains assertion on named individuals in terms of vocabulary. ABox has extensional knowledge about the domain of interest called membership assertion. A fact is an instance of a particular concept or role. For example, the facts a, b are assigned to concept C or roles R . Consequently, there are two types of assertions in ABox.

- Concept assertions: $C(a)$ means a belong to C . For example; $Person(Alan)$ shows that $Alan$ is a member of $Person$ class.
- Roles Assertions: $R(a, b)$ where a is connected to b using the role R . For example; $hasFever(Alan, High)$ means $Alan$ has $High$ fever. This property correlates two different concepts ($Patient$ and $Fever$) in order to express its meaning. The property $hasFever$ defines the relationship between $Patient$ and $Fever$ class and $Alan$ is an instance of a $Patient$ class and $High$ is an instance of a $Fever$ class.

3. **RBox (Relational Box)** reveals the inter-dependencies between the roles (r, s) . RBox includes a set of statements which describe the characteristics of roles such as symmetry, transitivity and reflexivity. The characteristics of RBox are expressed as a union of finite set of roles with the role hierarchy.

Concept Symbol	Concept Terms	Role Symbol	Role Terms
\top	Top	$R \sqcap S$	Role conjunction
\perp	Bottom	$R \sqcup S$	Role disjunction
$C \sqcap D$	Concept conjunction	$\neg R$	Role complement
$C \sqcup D$	Concept disjunction	$R \subseteq S$	Role inclusion
$\neg C$	Concept complement	$R = S$	Role equivalence
$\forall R.C$	Universal restriction	$R \circ S$	Roles composition
$\exists R.T$	Existential restriction (Limited quantifier)	$id(C)$	Identity role on C
$\exists R.C$	Existential Restriction		

Table 3.1: Description Logic Constructors

All DLs are based on the vocabulary consisting individuals, concepts and roles. The syntax of description logic languages allows users to construct the complex descriptions of DLs vocabulary. It also checks the satisfiability and consistency to establish the correct and meaningful knowledge base. Table [3.1] illustrates the constructors of description logic. Semantics deal with the meanings and interpretations. The knowledge base is interpreted in case of non-empty set of a domain. By considering a specific domain, concepts can be interpreted with the set of individuals. In a similar approach, roles are interpreted with the set of ordered pairs of individuals. In DLs, interpretation is normally denoted by I and a non-empty set is represented by Δ (also known as domain). Concept symbols are mapped to subset of the domain Δ and role symbols to subsets $\Delta \times \Delta$. Interpretation function I extends to complex concepts and roles. Suppose KB be the knowledge base consisting of a set of concepts and roles, so Δ^I satisfies KB which is written as: $\Delta^I \models KB$. This states that Δ^I satisfies every sentence in KB . So, $KB \models \alpha$ states that KB entails α if α is satisfied in the knowledge base KB .

3.4 The Semantic Web

The project discussed in this thesis aims to apply the semantic web technologies such as OWL 2 RL ontology and SWRL to construct the domain model knowledge base. For this purpose, we begin with the definition of semantic web and proceed with how it is vital

for our project. The semantic web concept was initially proposed by Tim Burners Lee [Daconta et al., 2004] with the intention of sharing knowledge and the semantics of the data beyond the scope of applications and WWW. The formal definition of the semantic web is given as:

“The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation” [Berners-Lee et al., 2001].

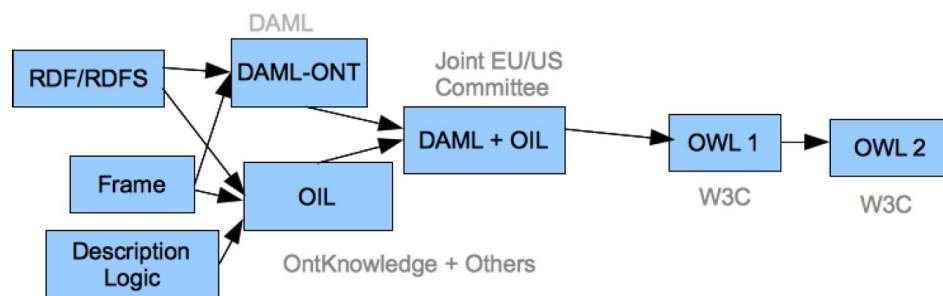


Figure 3.1: OWL Family Tree

The semantic web is a set of standards that can be used for RDF (resource description framework) data model, RDF schema, query language such as SPARQL and web ontology language (OWL) to store vocabularies and ontologies. RDF is the foundation of the semantic network, its syntax allows one to show concepts (classes) and properties (roles). RDF knowledge base contains triple (subject, predicate and object) which can be mapped to corresponding description logic axioms. For example, subject and object can be mapped to corresponding concept or individual whereas predicates to role. RDFS is an extension of RDF having sub-class and sub-property relationship [Gruber, , Faruqui, 2012]. The DARPA Agent Markup Language (DAML) is the next generation of RDF developed for frame-based knowledge representation and object-oriented languages. Initially, OIL (Ontology Inference Layer) was introduced to integrate the semantic web based core elements from frame languages, description logics, XML and RDF. OIL is based on

DL SHIQ. Soon later in 2002, a new language was developed by joint venture of DAML and OIL, named as DAML+OIL. Basically, Description Logic based model is technically influenced by its formal semantics. In DAML+OIL, description logic inferred language constructors have been preserved but their frame structure was totally changed for the integration of RDF syntax [Hendler and McGuinness, 2000, Fensel et al., 2001, Horrocks et al., 2002]. The OWL family tree [Bechhofer, 2007] is shown in Figure 3.1, which depicts how OWL has developed from the semantic web technologies.

FOL	Description Logic	OWL
Unary predicate	Concept name	Class name
Formula with one free variable	Concept	Class
Binary predicate	Role name	Object property name
Formula with two free variables	Role	Object property
Theory	Knowledge base	Ontology
Sentence	Axiom	Axiom
Signature	Vocabulary/signature	Vocabulary

Table 3.2: Corresponding Terminologies of FOL, DL and OWL

The design of OWL (Web Ontology Language) was influenced by DAML+OIL, DL, RDF and RDFS. The OWL (Web ontology language) depends on description logic languages which provides extra sort of additional logical information, for example; ontology versioning and annotations. OWL has a support for developing and reasoning ontology with their data types. DLs knowledge base satisfiability can be checked by using the reasoning tools in the OWL. Thus DL and OWL have a very strong association for modelling logical frameworks and verifying those using reasoning tools. Description Logic knowledge base translation into OWL is straightforward. In addition, DLs axioms can be translated to FOL corresponding axioms. Table 3.2 illustrates the summary of corresponding terminologies used in FOL, DLs and OWL which is extracted from [Rudolph, 2011]. More specifically, some essential components are required to translate DL SROIQ knowledge base into OWL. First, Preamble contains the definition of the namespace. Second, declaration of the used concept and role names should be according to the class and object proper-

ties respectively. Third, OWL axioms should be from the DL knowledge base [Rudolph, 2011].

The best practice for accessing and sharing knowledge is to use RDF data model, OWL ontologies with their unique identity known as URI (universal resource identifier) to assign unique name to each individual. The semantics of data usually defines the meaning of the data, for this purpose, W3C (World Wide Web consortium) lets the web ontology language (OWL) to store meaningful data with their URIs. In the semantic web, every single source or content of the source has a unique identifier. The most common identifiers are URLs, URNs and URIs. The first one is the URL (uniform resource locator) which is used to identify specific set of resource. For example; <http://www.nottingham.edu.my>. Second, URN (Universal resource Name) can be accessed using unique id numbers. Third, URI (universal resource identifier) is a superset of the URLs and the URNs which includes both identifiers. URI can be used to access all contents of the resources and URL is itself a subset of URI.

3.4.1 Web Ontology Language (OWL)

OWL (also known as OWL 1.0) is a computation logic based on formal language for representing ontologies. OWL 1.0 is based on description logic language SHION that provides the syntax and the semantics to represent ontologies. OWL Ontology includes a set of axioms that provides logical assertion about three core elements: classes, object, data properties and individuals. OWL 1 has three sub-languages [Van Harmelen and McGuinness, 2004]. First, OWL Lite provides concept hierarchies and property restrictions. It corresponds to SHIF that provides support to those users who essentially require simple constraints and classification hierarchy. Second, OWL DL provides supports for the users who require utmost expressiveness, computational completeness and decidability. Third, OWL Full is specifically designed for the users who require utmost expressiveness with RDF syntactic freedom but it does not have computational guarantees. OWL full permits

ontologies to extend the meanings of the RDF and/or OWL predefined vocabularies.

3.4.2 OWL 2.0

OWL 2.0, an extended version of OWL 1.0, formally describes the ontology languages with their defined meanings for the semantic web. It is based on description logic language SROIQ to represent ontologies. Accordingly, OWL 2 is more expressive than OWL 1, and yet it has intractably combined the computational and data complexities while performing the reasoning problems. OWL 2 ontologies specify classes, properties, individuals, and data values. The core purpose of OWL 2 is to capture knowledge and model the language for specific domains about the human knowledge. In OWL 2, ontology consists of a set of statements (classes and properties with their defined relationship) and a set of rules. These statements are called axioms in OWL 2 [Hitzler et al., 2009]. There are different profiles in OWL 2, each profile has certain restrictions according to the scenario for ontology structures and reasoning tasks.

- OWL 2 EL is useful for those applications employing ontologies which consists of numerous classes and properties. It is typically intended for the applications that use ontologies with large volume of TBoxes.
- OWL 2 QL is specifically designed for applications that utilize big amount of instance data in which query answering is a crucial reasoning task. It is suitable for developing ontologies with large size of ABoxes.
- OWL 2 RL reasoning systems permits rule-based reasoning. It has a higher expressivity as compared to OWL 2 EL and OWL 2 QL. It requires scalable reasoning with less expressive power and can be implemented using rule-based reasoning engine. OWL 2 RL is designed to accommodate both OWL 2 and RDF applications. OWL 2 applications deal with the maximum expressiveness of the language of efficiency while RDF (S) applications require additional expressiveness for OWL 2.

Property axioms define many relations about properties. These properties can be transitive, symmetric and asymmetric

3.4.2.1 Why to Choose OWL 2 RL

OWL 2 RL Axioms Assertion Axiom	DL Axioms
ClassAssertions	$C(a)$
ObjectPropertyAssertion	$P(a, b)$
DataPropertyAssertion	$P(a, b)$
SubClassOf	$C \sqsubseteq D$
EquivalentClassOf	$C \equiv D$
OWL 2 RL Object Property Axiom	
SubObjectPropertyOf	$P \sqsubseteq Q$
ObjectPropertyChain	$P \circ Q \sqsubseteq R$
EquivalentObjectPropertyOf	$P \equiv Q$
InverseObjectPropertyOf	$P = Q^{-}$
ObjectPropertyDomain	$T \sqsubseteq \forall P^{-}.C$
ObjectPropertyRange	$T \sqsubseteq \forall P.C$
SymmetricObjectProperty	$P = P^{-}$
TransitiveObjectProperty	$(P \circ P \sqsubseteq P)$
OWL 2 RL Class Expression	
ObjectUnionOf	$C_1 \sqcup C_2 \sqsubseteq D$ or $C \sqsubseteq D_1 \sqcup D_2$
ObjectIntersectionOf	$C_1 \sqcap C_2 \sqsubseteq D$ or $C \sqsubseteq D_1 \sqcap D_2$
OWL 2 RL Class Expression (Property Restriction)	
ObjectAllValuesFrom	$C \sqsubseteq \forall P.D$
ObjectSomeValuesFrom	$\exists P.C \sqsubseteq D$

Table 3.3: OWL 2 RL Axioms

We have decided to use OWL 2 RL based ontology to model domains for our logical frameworks because, in this thesis, context-aware agents use rule-based reasoning techniques. OWL 2 RL ontology can be used to define the contextual knowledge in terms of concepts and roles. On the other hand, SWRL rules can be used to define more complex rules using OWL concepts and roles. Hence, both OWL 2 RL and SWRL rules are integrated in the form of knowledge base which is used by context-aware agents. In fact, OWL 2 RL profile is specifically designed to implement rule-based reasoning systems. A set of OWL 2 RL ontology axioms can be translated into a set of Horn-clause rules [O'Connor and Das, 2012, Grosz et al., 2003]. OWL 2 RL profile depends upon scalable reasoning

systems deprived of losing the expressive power which can be implemented using rule-based reasoning engine. Table 3.3 shows OWL 2 RL axioms with their corresponding mapping to description logic axioms.

3.4.3 Why OWL is Not Enough?

OWL 1 is not able to express the chain relations such as

hasDoctor(?x, ?y), *EmployeeOf*(?y, ?z) \rightarrow *DiagosedAt*(?x, ?z)

We can express this relation using property chains in OWL 2 [Krötzsch et al., 2011]. However, complex rules cannot be expressible in OWL 2 [Vilasrao and Bhaskar, 2012]. For example, *Person*(?p), *BloodSugarLevel*(?bsl), *hasBloodSugarLevelBeforeMeal*(?p, ?bsl), *greaterThan*(?bsl, '126) \rightarrow *hasDBCcategory*(?p, 'EstablishedDiabetes).

We need to write complex rules using SWRL to address this kinds of issue. Moreover, OWL 2 is a declarative language which describes the state of affairs in a logical way. In addition, DL-safe rules have been introduced which are implemented using reasoner. Reasoning tools can be used to derive information about the state of affairs.

3.4.4 Semantic Web Rule Language (SWRL)

SWRL is essentially based on OWL rule languages which stipulates the power to write down Horn like rules in terms of OWL concepts and roles. It has a sound reasoning capability with the OWL. As OWL languages are not always able to express all kinds of relations, however, this can be done using SWRL. In ontology, OWL 2 RL rules can be written using class with the sub-class relationship and property with the sub-property relationship. The combination of OWL 2 RL and SWRL provides better expressive power as compared to OWL 2. Protégé ontology editor [Protégé, 2011] has a built-in feature that supports SWRL rules and therefore the reasoners like Hermit [Motik et al., 2009], Pellet [Sirin et al., 2007], Fact++ [Tsarkov and Horrocks, 2006], RacerPro [Haarslev and Möller, 2001], Kaon2 [Motik and Studer, 2005] etc. also support SWRL rules. These

rules enhance the expressivity of the OWL ontology languages. SWRL rules encompass class atoms, object property and data property atoms. Class Atoms consist of OWL named class or class expression having a single argument while the properties consist of OWL object property or data valued property with two arguments where data values may be of different data types.

SWRL rules are of the form of antecedent-consequent pair in which antecedents are sometimes in the form of conjunctive pairs. The antecedents of the rule refer to the body whereas consequent refers to the head of the rule. The formulation of the rules with the implication is given as:

$$atom_1, atom_2, \dots, atom_n \rightarrow atom.$$

where the left side of the arrow (\rightarrow) represents the body, whereas the right side represents the head. The consequent (head) of the rule will be true whenever all atoms (classes and/or properties) are true in the antecedent (body) of the rule. Rules with conjunctive atoms in the consequences are reconstructed into multiple rules. Each reconstructed rule must have an atomic consequent after segregation. These rules have the reasoning capability to infer OWL individuals with regard to OWL classes and properties [connor et al, 2005].

In [Horrocks et al., 2004, Kuba, 2012], different SWRL declarations (atom types) are given for representing class and property atoms. Class atoms consist of OWL named classes with a single argument ($Person(?p)$). Individual property contains OWL object property with two arguments ($hasBrother(?x, ?y)$), data valued property allows OWL data property with two arguments ($hasBrotherName(?x, "Alan")$) where data values may be of different data types. Different individual atoms are expressed as “differentFrom” with two arguments. For instance, let us say x is different from y , so it is represented as $differentFrom(?x, ?y)$. The same individual atoms are represented using “sameAs” clause having two arguments. For this, let us consider x is similar to y , so it is represented as $sameAs(?x, ?y)$. Data Range contains individual, property variable of particular type, for example; $xsd : int(?x)$. Sometime data range express one of the re-

lationship like $\{3,4,5\}(?x)$. Built-Ins is a very useful feature to write down customized individuals for SWRL rules. Built-ins can be used in the antecedent of the rule with its namespace qualifier. It accepts one or more parameters and evaluate as true if parameters satisfy the predicate.

SWRL rule does not support Negation as Failure (NaF); however, classical negation is supported for writing SWRL rules. For example, the rule

$$\neg Person(?x) \rightarrow \neg Human(?x)$$

is not possible in the ontology. Protégé ontology editor [Protégé, 2011] does not allow to write this rule but it can be written in the following way:

$$(not(Person))(?x) \rightarrow nonHuman(?x)$$

or
$$(not(Person))(?x) \rightarrow (not(Human))(?x)$$

If OWL classes are disjoint then rules can be safely concluded, for example, *Man* and *Woman* are two disjoint classes. SWRL rules support only positive conjunctive atoms and does not support disjunction of atoms.

The semantics of SWRL are OWL DL based which does not have a direct support for reasoning about classes and properties. SWRL built-in features are compatible with OWL classes, object and data properties that allow to write complex rules. SWRL built-in features also provide support for RDF and RDFS ontologies, which can be converted into OWL ontology and their constructs can be mapped to their corresponding OWL constructs.

SWRL supports monotonic inference only. SWRL rules can not be used to modify existing knowledge in the ontology. Non-monotonicity may occur in case if modifications made on SWRL rules. SWRL rules can not be used to revise the existing information from ontology. For example,

$$Person(?p), BodyTemperature(?temp), hasBodyTemperature(?p, ?temp), \\ greaterThan(?temp, '99) \rightarrow hasFever(?p, 'Yes).$$

The result will add value to “Yes” only if fever is above “99” otherwise false.

3.4.4.1 DL Safe Rule in SWRL

The essential drive of the DL Safe rule is decidability. DL-Safe rules are restricted subset of SWRL rules in which decidability is gained from the set of rules. Decidability is guaranteed only for the class and property atoms in ontology. Alternately saying, the set of all variables in DL Safe rules is considered with those individuals only which are known in the ontology. SWRL rules do not stand alone, these rules work together with the OWL ontology axioms while the reasoning is performed. In the following, we provide a simple SWRL rule to show how a SWRL rule can be transformed in a functional syntax, i.e., DL safe rule.

$$Person(?p), PatientIdentification(?pid), hasPatientID(?p, ?pid) \\ \rightarrow Patient(?p).$$

```
Prefix(var :=< urn : swrl# >)
Declaration( Class( : Patient ))
SubClassOf( : Patient : Person )
DLSafeRule(
  Body(
    ClassAtom( : Person Variable(var : p))
    ClassAtom( : PatientIdentification Variable(var : pid))
    ObjectPropertyAtom( : hasPatientID(var : p) Variable(var : pid))
  )
  Head(
    ClassAtom( : Patient Variable(var : p))
  )
)
```

The core purpose of reviewing literature on OWL 2 RL and SWRL is to realize their significance in modelling domains of human knowledge and develop rules according to the case study. In Chapter 6, we present a tool which extracts these rules from an ontology

(OWL 2 RL and SWRL) and then translate them into a set of plain text Horn-clause rules. In the next section, we describe rule-based system and how rule-based reasoning works considering context-aware reasoning agents.

3.5 Rule-based System

Rule-based system is an automated reasoning technique which captures and refines human expertise. It automatically performs reasoning for problem solving [Hayes-Roth, 1985]. A bulk of research on rule-based systems has been considered for decades in different areas of computer science. These have played very influential roles in various areas of computing systems including semantic web [Yang et al., 2003, Shadbolt et al., 2006], sensor networks [Dressler et al., 2009, Terfloth et al., 2006], context-aware systems [Hong et al., 2009, Perera et al., 2014], and business process modelling [Lu and Sadiq, 2007] etc. In general, rule-based systems capture data, analyze data based on the available information and then perform reasoning to produce the desired goals. A rule-based system is actually an inference based technique that has flexible and dynamic implementation mechanism to solve more realistic and complex problems. However, this system is now becoming much more challenging in AI, predominantly in distributed systems, where systems exchange information among components (e.g., agents in our case) via messages.

Literature highlighted several rule-based agents' frameworks; for example, [Alechina et al., 2006, Jago, 2009, Daniele et al., 2007, Alechina et al., 2009a]. On the semantic web, applications of rule-based systems are used for ontology based reasoning [Qin et al., 2007, Rakib et al., 2012], more specifically, OWL 2 Rule Language (OWL 2 RL) [Ter Horst, 2005], Semantic web rule language (SWRL) [Horrocks et al., 2004], and combination of Rule ML and Web Ontology Language (OWL). These kinds of reasoning enhance the expressive power for underlying ontology languages. Rule-based system has also played vital roles for modelling business domains [Grosz and Poon, 2003, Friedman, 2003]. For

example, business rules define all legal constraints for the business domains. In other areas of AI, rule-based system is used to monitor the behaviour of expert systems.

3.5.1 Components of Rule-based System

Rule based system has four components:

1. Rule-base (Knowledge Base): specifies particular type of knowledge bases of the domain in terms of rules where knowledge is encoded into IF-Then rule form.
2. Temporary Working Memory: stores the facts used by inference engine. A fact may be a sensor reading for context-aware rule-based systems.
3. An Inference Engine: performs reasoning to derive new information and/or takes appropriate actions based on the input and rule base. Match-resolve-act has the following phases:
 - Match: It is the first phase in which facts on the left hand side of the rule (or conditions, i.e IF) are matched with the contents of the working memory. If facts on the left hand side of the rules are matched then conflict set is generated. This constitutes an instantiation of the rules. In this phase, one rule may have more than one instantiations. Conflict set is generated for the set of all rule instantiations in the memory.
 - Conflict Resolution: In this phase, conflict resolution strategy selects a single instance or a subset of conflict set for execution from the conflicting set. The set of all instantiations are chosen for the execution in case when reasoning strategy is not applied.
 - Act: After choosing the single or multiple instantiation(s) from the conflict set, the rule will be executed to infer new information. These actions (newly derived information) are added to the working memory or this can be overwritten (as in our case where memory is bounded). This phase will move to

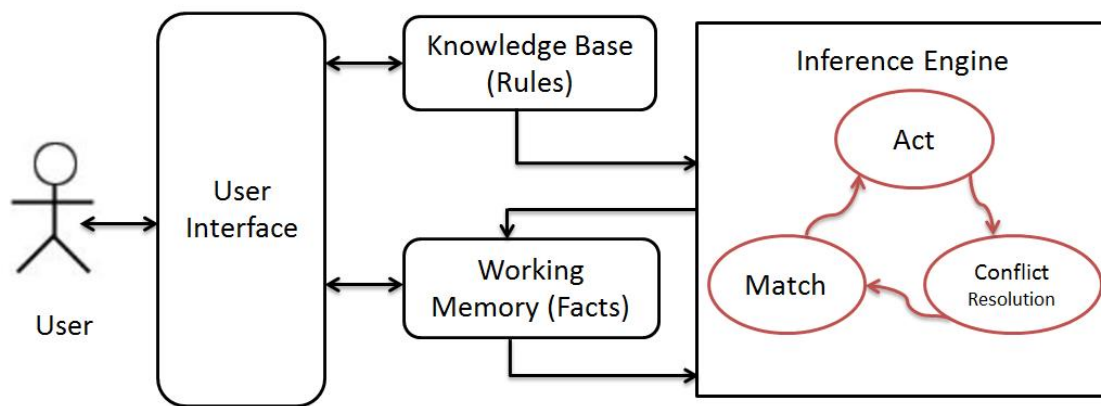


Figure 3.2: Structure of Rule-based System

the first phase of match-resolve-act cycle and continues until and unless no more rules are matched.

4. User Interface: This provides a platform through which input/output signals are sent and received. User interface is optional part of the reasoning process, but it is recommended.

3.5.2 Multi-agent Rule-based System

Rule-based system has gained significant attention in multi-agent systems due to its great degree of abstraction to specify behavior of agents. In multi-agent reasoning systems, agents use different reasoning techniques such as Resolution based reasoning [Robinson, 1965]; Modus Ponens and conjunction introductions [Walter Sinnott-Armstrong, 1986]; and rule-based reasoning [Hayes-Roth, 1985] etc. In rule-based multi-agent system, each agent has a unique name and its own program which perform reasoning on the same principle of rule-based system. Rule-based agents have a program consisting of a knowledge base and a working memory. Knowledge base consists of a set of rules whereas working memory contains the set of facts used by rules. The generic representation of a rule is given as:

IF *<Statements to be evaluated>* → THEN *<Actions>*

where the left hand side of the arrow contains the set of conditions to be evaluated. When these conditions are matched, the rule will be executed to generate *THEN* (consequent) part. We follow the similar approach for context-aware rule-based agents in this thesis.

Rule-based multi-agent systems may have hundreds of rules to model the domain of complex case scenarios. When the size of the domain is very large and complex, there is a need to apply some reasoning strategies. For example, some strategies are needed to be applied only on the applicable rules in order to avoid putting on all matching rules instances. In MAS rule-based system, each rule is given a rule priority in accordance to its significance in the system. For example, the most important rule is given the highest priority.

There are several benefits for adopting rule-based approaches with the formal methods [Jago, 2006] to establish properties of the resulting systems with respect to:

- **Correctness:** This is a measure to check whether a rule-based agent would be able to produce the correct output against all legal inputs or not.
- **Response Time:** This calculates the computational time steps taken by a rule-based agent to generate any output.
- **Termination:** This determines whether a rule-based agent would be able to produce an output at all or not.

3.6 Formal Approaches for Distributed MAS

This section emphasizes how basic logics are interlinked with ontology languages in order to design logical frameworks. We also discuss the ontology-based context-aware frameworks incorporating resource-bounds.

3.6.1 Ontology-Based Context-aware Systems

Recent developments in the field of context-aware systems have heightened the significance of smart environment applications using different smart (resource-limited) devices. Literature [Esposito et al., 2008, Wang et al., 2004b, Rakib and Faruqui, 2013, Ejigu et al., 2007] reveals a large volume of context-aware systems which are based on ontology due to its formal knowledge representation structure. According to [Ejigu et al., 2007], an ontology-based generic context management (GCoMM) model was presented to facilitate context reasoning by providing ontology based structure for context instances, rules including their semantics. The contexts and their semantics are expressed using upper and lower level ontology in the GCoMM model. It is semantically a rich model that supports collaborative reasoning for multi-domain context-aware applications. Their rules are either derived or defined using ontology compatible rule languages according to the requirements of the application domains. In this case, resource limitation is a key issue. Esposito et al. proposed an ontology-based context-aware computing framework [Esposito et al., 2008] with its prototypal implementation by considering a home care case study. This framework is based on ontology which facilitates contexts codification to support agents' reasoning and communication. This system combines its multi-agent behavior and synchronization of heterogeneous technologies such as ontologies, multi agents, and rule-based inference engines. In this system, ontology provides contextual information (vocabulary) to agents to perform rule based reasoning. This enables them to infer results which can be exchanged based on agents interoperability. This framework has been implemented using a simple home care scenario using coordinated operations and efficient interoperability of ontology driven knowledge base, agents with their rule-based reasoning.

Wang et al. [Wang et al., 2004b] have presented an ontology based formal context model with the intention of handling critical matters including context representation, logic-based contextual reasoning and knowledge sharing. In the model of logic based contextual

reasoning, the detailed investigations have been performed on the feasibility of contextual reasoning in pervasive computing environments. In a pervasive computing environment, this prototype quantitatively assesses the feasibility of logic-based context reasoning for critical applications where limitation of computational resources are carefully monitored.

3.7 Monotonic Vs Non-monotonic Reasoning

Monotonic reasoning is a well-known reasoning technique that is used in various traditional logics and applications. It is known as static knowledge that entails monotonic reasoning based on open world assumptions and takes surety of propagation of truths. The general truth of statements such as factual or ontological knowledge does not change when new information is added. In monotonic reasoning system, there is no need to check inconsistencies because previously known information can never become invalid after adding the new information. Monotonic reasoning does not deal with inconsistent and incomplete information. Most suitable examples of monotonic reasoning are the traditional systems with predicate logics [Nute, 2003, McDermott and Doyle, 1980].

Monotonic Logics lack the phenomenon of acquiring new information with the replacement of former knowledge whereas non-monotonic reasoning provides a concrete solution to resolve this issue. Accordingly, non-monotonic reasoning is based on commonsense reasoning. It connotes that humans often draw some conclusions based on partially known information and then retract or revise conclusion based on the availability of correct and complete information. In non-monotonic reasoning, the recently derived conclusion may also be revised based on the contrary evidence from the rules of inference. It is also known as dynamic knowledge which is more flexible and inconsistency tolerant non-monotonic reasoning. In this regard, it is more suitable for deriving conclusions. In a highly dynamic environment, knowledge is frequently updated with every possible change in the current situation.

Recently, there has been increasing interests in developing non-monotonic reasoning based formalisms [Grosz et al., 2003, Antoniou, 2002, Nute, 2003] to deal with inconsistent and incomplete information. Due to incomplete information, conclusion is drawn based on some assumption that is supported by classical predicate logic. Conflicts among rules are resolved implicitly or explicitly based on the availability of priority information. Non-monotonic rule systems can be used to prioritize information based on contradictory information and is widely used in ontology languages. This kind of reasoning is very close to human because human reasoning is usually non-monotonic. A person often changes his mind and rejects his/her own decisions based on new evidence, even though, these decisions were justified by his own at some previous time. On the other hand, medical diagnosis is an excellent example of non-monotonic reasoning system where expert decisions or inferences are quickly taken to draw defeasible inferences. More so, non-monotonic logics are very useful in these systems to deal with inconsistent and incomplete information in a declarative way.

Literature has revealed several non-monotonic reasoning techniques [Grosz, 1997, Antoniou et al., 1999b, Reiter, 1980, Kakas et al., 1992, Paul, 1993, McDermott and Doyle, 1980, McCarthy, 1987] to deal with defeasible conclusion. In 1980, Default logic was proposed by Raymond Reiter [Reiter, 1980] to formalize reasoning where possible hypothesis are true by default. It describes the process of jumping to conclusion based on certain assumptions. Default logic expresses the known facts when conclusion is true by default in the absence of conflicting information in which rules are modelled and priority relation is defined among them. Another well-known non-monotonic reasoning formalism is Courteous logic [Grosz, 1997, Antoniou et al., 1999b] which not only deals with classical negation, but also considers prioritized conflict handling by comparing the set of rules. The specific feature of courteous logic is its atom dependency graph which is acyclic. Unlike defeasible logic, courteous logic program does not distinguish between strict conclusions and defeasible conclusions; however, this program may use Negation as

Failure (NaF). Abductive logic [Kakas et al., 1992, Paul, 1993] is based on non-monotonic reasoning formalism in which derivation models explanations of the known facts which are not necessarily true. More precisely, Abductive reasoning generates the casual explanation of the known facts which are incorrect. Applications of Abductive reasoning could be suitable for model-based scenarios; for example, medical diagnosis. This logic is normally used to interpret ambiguous statements. Another non-monotonic reasoning approach is the autoepistemic logic, introduced by Moore [Moore, 1985, Creignou et al., 2012]. This is an extension of propositional logic by a unary modal operator L which states that its argument is believed. It was proposed to overcome difficulties of other formalisms such as non-monotonic modal logic [McDermott and Doyle, 1980], default logic [Reiter, 1980] and circumscription [McCarthy, 1987]. The abstract idea of autoepistemic logic is its consistent belief sets from the set of premises which is known as stable expansions defined by consequences of the premises and beliefs. Belief revision is also one of non-monotonic reasoning techniques to revise its belief based on one new contradictory belief to avoid inconsistency [Alechina et al., 2008].

3.8 Defeasible Reasoning

In recent years, defeasible reasoning has been considered as one of the most successful sub-area in non-monotonic reasoning to deal with inconsistent and incomplete information due to low computational complexity and its focus on implementability [Antoniou, 2002]. Defeasible reasoning is a simplistic rule-based technique used to reason partial and conflicting information. This is a rule-based approach without Negation as Failure (NaF). Rules consist of first order atoms (facts) which are interpreted defeasibly. Accordingly, priorities can be used to resolve conflicts among rules based on the knowledge. Defeasible rules are prioritized and can be defeated by contrary evidence. In defeasible reasoning, sceptical approach does not allow contradictory conclusion to be drawn.

3.8.1 Defeasible Logic

Defeasible Logic is one of the most promising and fine-grained non-monotonic reasoning formalisms to deal with the inconsistent and incomplete information. Defeasible logic was originally introduced by Nute [Nute, 2003] to address certain aspects of reasoning with the particular concerns to improve its computational efficiency without considering Negation as Failure. It is a simple but efficient rule-based reasoning technique that derives plausible conclusions from conflicting knowledge while preserving low computational complexity [Maher, 2001, Lam, 2012, Maher et al., 2001]. Defeasible logic provides greater flexibility in terms of its expression of information. The core purpose of using defeasible logic is to produce plausible conclusions by modelling a situation where conflicting rules appear concurrently. This logic deals with potential conflicts among knowledge items and has classical negations contrary to usual logic programming system. Potential conflicts are resolved by superiority relation. Defeasible logic is known as sceptical non-monotonic formalism which supports both kinds of reasoning monotonic as well as non-monotonic.

There are five essential components in the defeasible logic. We briefly define these components in this section, however their examples with illustrations are given in Chapter 5.

1. **Facts** are indisputable and undeniable statements. For example; a fact '*Alan is a Person*' is formally written as $Person(Alan)$.
2. **Strict Rules** are very similar to the rules used by classical logics which state that whenever premises of the rule are indisputable then the conclusion is obvious. The conclusion (newly derived fact) is straightforward and unquestionable. These rules can never be defeated. These rules are of the form: $P_1, P_2, \dots, P_n \rightarrow P$
3. **Defeasible Rules** are the rules that can be defeated by contradictory conclusion. These rules are of the form: $P_1, P_2, \dots, P_n \Rightarrow P$

4. **Defeaters** are the rules that can be used to prevent the derivation of contradictory conclusion. These rules are designed not to draw conclusion but to block conclusion. In other words, defeater rules defeat defeasible rules based on conflicting information. These rules are of the form: $P_1, P_2, \dots, P_n \rightsquigarrow P$.
5. **Superiority Relation** (\succ) is a very good feature of defeasible logic having binary relation between rules to determine the strength of rules and then prioritizes superior rule to be fired. Superiority relation resolves conflicts among rules by prioritizing rules when more than one rules instances are concurrently fired to draw conclusion. Superiority relation cannot be used for strict rules. Strict rules are always superior to defeasible rules.

In recent years, Defeasible logic has gained significant attention in non-monotonic reasoning community theoretically as well as practically. In theoretical aspects, literature has revealed several studies including proof theoretic [Maher, 2002], proof theory [Antoniou et al., 2001], argumentation semantics [Governatori et al., 2004], including temporal aspects such as temporal defeasible reasoning [Augusto and Simari, 2001, Governatori et al., 2007] and Normative position [Governatori et al., 2005]. The applications that delve with defeasible logic has various domains, including, semantic web [Kravari et al., 2010, Bassiliades et al., 2004, Governatori and Pham Hoang, 2005, Antoniou and Bikakis, 2007], modelling business rules [Antoniou et al., 1999a], agent modelling and negotiation [Dumas et al., 2002, Governatori and Rotolo, 2004, Dastani et al., 2005], legal reasoning [Grosz et al., 1999], modelling of contracts [Governatori, 2005]. In this thesis, we consider defeasible logic to develop the logical framework $\mathcal{L}_{\mathcal{DROCS}}$ based on semantic web technologies (OWL 2 RL and SWRL) following rule-based technique [Esposito et al., 2008, Daniele et al., 2007, Gómez et al., 2007] for context-aware non-monotonic reasoning agents.

3.9 Defeasible Reasoning based Distributed Systems

As mentioned earlier, defeasible reasoning is one of the most prominent and successful sub-area in non-monotonic reasoning to reason conflicting information using rule-based reasoning. Due to its simple and flexible nature, defeasible logic has attracted significant interest of the research community towards various domains, particularly in knowledge representation and their reasoning in multi-agent systems. In this section, we briefly survey some ontology driven defeasible reasoning based formalisms which include rule-based reasoning technique.

3.9.1 Defeasible Reasoning based Frameworks for the Semantic Web

Recent developments in the field of non-monotonic reasoning led to a renewed interest in the semantic web. The idea of defeasible reasoning based techniques for the semantic web applications was initially proposed by Antonis Bikakis and Grigoris Antoniou. Afterward, research in this area got acceleration towards different domains including medical, business, brokering systems etc [Antoniou and Bikakis, 2007, Skylogiannis et al., 2007, Antoniou et al., 2005]. However, ontology is one of the essential components of the semantic web. This is owing to the fact that mostly semantic web based approaches are based on ontology due to its knowledge representation using OWL and RDF formats. In 2005, Antoniou et al. [Antoniou et al., 2005] have proposed semantic-based e-brokering system that allows the service requesters (agents) and service providers (agents) to match their interests against their offers congregated by a broker agent. This system uses semantic web RDF standards to represent the set of offerings, and a deductive logical language is used to present the requesters' requirements and their preferences. The ultimate goal is to identify suitable services to satisfy users' requirements and choose the best service from user preferences. Authors have selected defeasible logic to represent requesters' requirements and preferences due to its expressiveness. Defeasible rules specify priorities to show user preferences from most appropriate offerings. The service requesters' require-

ments are expressed in the logical language using rules and priorities. The set of offerings are represented in RDF statements to be shown in web resources. RDF statements are transformed to logical facts and rules using RDF translator. The rule translator translates the rules (submitted by requester) into Prolog rules that compete with the semantics of defeasible logic. JADE¹ is a Java based open source middle-ware framework which has been used to develop this system.

An automated agent negotiation system [Skylogiannis et al., 2007] is designed to capture the behavior of parties (sender and receiver agents) involved in the negotiation. Its main behavior is to automate negotiation process in e-commerce system. Accordingly, an automated negotiation system is a process in which two or more agents communicate with each other and then finally they agree on one acceptable decision. This system uses JADE agent framework and is based on an executable formal approach. Declarative negotiation strategy is one of the most distinctive features to be expressed in a declarative rule language (defeasible logic). Defeasible logic has been considered as one of the most promising solutions for brokering preferences modelling and negotiation strategies. These negotiation strategies are applied using a system DR-DEVICE. As stated in [Bassiliades et al., 2004], DR-DEVICE is an implemented defeasible reasoning system for reasoning on the web. This kind of reasoning is very useful for ontology integration where conflicts among rules are derived on the semantic web. On the other hand, defeasible logic rules are used to reason RDF data over multiple web sources. The system interface is compatible with RuleML. This is based on CLIP production rules system [Bassiliades and Vlahavas, 2003]. Defeasible knowledge has been translated into a set of deductive rules with the aggregate and derived attributes. This system has two major components: rule translator and RDF translator. The rule translator takes rules from the system (user) and then translates them into CLIPS production rule system. Defeasible logic rules are first translated into a set of deductive rules and then further translated CLIPS production rules.

¹<http://jade.cselt.it/>

Antoniou et al. [Antoniou and Bikakis, 2007] have proposed a defeasible reasoning based declarative system for the semantic web. This system is very flexible in a sense that it is compatible with semantic web standards such as RDFS, OWL, RuleML and Prolog. The implemented system is prolog based and can reason with both monotonic as well as non-monotonic rules while ontological knowledge domain is built in OWL or RDFS format. Defeasible reasoning technique has chosen to resolve conflicts among rules. In this framework, defeasible knowledge can be transformed into logic programs with declarative semantics. DR-Prolog supports both monotonic as well as non-monotonic rules to deal with inconsistencies. However, this model can be extended with agent-based system.

3.9.2 Integration of Description Logics with Defeasible Reasoning

Defeasible reasoning is simple and efficient. It makes sense to integrate defeasible reasoning with description logic as both share focus on efficiency. The two core reasons of integrating description logics with defeasible reasoning are:

- Enhanced reasoning capabilities can be represented in much broader ontological knowledge.
- Rule-based system defines ontology-based applications using vocabulary expressed in description logic.

The incorporation of defeasible reasoning on the top of description logic acquired considerable attention in non-monotonic reasoning paradigms. Antoniou [Antoniou, 2002] has presented a framework for the integration of description logic with defeasible reasoning. In this system, Horn rules are conjoined with description logic which is a subset of predicate logic, but non-monotonic rules are not subset of predicate logic. The ontological knowledge is defined in description logic languages, and concepts and roles predicates are used in the antecedents of the rules but not in the head. Proof theory method has

been defined to deduce certain properties using defeasible reasoning. Although defeasible reasoning with description logic are interweaved to define the proof theory, their focus are limited to show rule system on the top of ontology. The above finding is consistent with the study mentioned in [Wang et al., 2004a] in which Wang et al. have proposed Description Defeasible logic (DDL). Accordingly, the combination of Description logics with the defeasible logic allows rules to be constructed on the top of ontologies. This framework includes description defeasible logic rules (ddl-rules) which represent the set of rules while a knowledge base represents the proof theory in description logics. These rules are prioritized based on superiority relation and defeasible rules. Defeasible rules, on the other hand, are used to defeat contrary evidence. The defeasible logic theory contains queries to represent DL knowledge base. This approach is flexible enough to build rules on the top of ontologies and vice versa in certain situations. The distinctive feature is that DDL is tractable like most of the description logic languages.

3.9.3 Integrating Rules and Ontologies using DeLP

In this section, we discuss how and why we combine rules with ontologies. The conjunction of rule languages and ontology driven knowledge has strongly influenced on various application domains in different aspects in the semantic web. Both ontology driven knowledge and rules are conjoined to produce desired results in a knowledge based form that can be used for different application domains and their applications.

Gómez et al. [Gómez et al., 2010] have proposed Defeasible Logic Programming (DeLP) for reasoning with inconsistent ontologies by integrating rules and ontologies which are suitable for extending current standard SWRL for representing rules on the top of the ontologies. This framework characterizes the behavior incorporating classical literals and negated literals in rules. The ontologies and rules are interpreted as DeLP that allow rules to reason on the top of a set of possibly inconsistent ontology. Description logic reasoner cannot perform reasoning with inconsistent ontology, for example, RACER [Haarslev and

Möller, 2001]. The DL reasoner is unable to extract useful consequences from inconsistent ontology definition. The inconsistent ontological knowledge can be used to perform reasoning not with the traditional reasoners, but with other reasoning techniques such as rule-based. With this, there is a need to translate ontology axioms into their corresponding DL axioms as subset of DLs. In the long run, it can be translated to their equivalent subset of Horn-logic. Defeasible logic programming is based on logic programming that has the capability of dealing with possibly inconsistent ontology definition which is codified as a set of Horn-clause rules.

3.9.4 Defeasible Reasoning based Multi-context Systems

Multi-context system encompasses several contexts which are interlinked with bridge rules to allow adding knowledge into a context depending on knowledge in other contexts. Multi-context system has applications in various areas, such as argumentation, data integration, or multi-agent systems. In [Bikakis et al., 2008], Bikakis et al. have presented a distributed approach for reasoning in ambient computing environment. Knowledge representation model based on the Multi-context system paradigm uses two kinds of rules as peer rules for local contextual knowledge and mapping rules (defeasible rules) through which ambient agents exchange information. Non-monotonic features were included in Multi-context system to resolve potential inconsistencies of distributed contextual knowledge. In this framework, mapping rules are modelled as defeasible rules that use context information to defeat contradictory information based on the fixed priorities for resolving conflicts. Four different conflict resolution strategies were used through which ambient agents exchange context in order to evaluate the quality of the imported context information. These strategies have been implemented for distributed reasoning algorithms to evaluate query in Multi-context system.

3.9.5 Discussion

Literature has revealed significant applications in practice and various researchers have proposed different approaches of defining a knowledge base as a pair of ontology and a set of rules including works by [Eiter et al., 2011, Levy and Rousset, 1998, Motik et al., 2005, Rosati, 2006]. However, the approaches proposed by [Gómez et al., 2007, Grosz et al., 2003] have mostly influenced the work presented in this thesis Chapter 5. In [Grosz et al., 2003] Grosz et al. have shown that the ontology based modelling techniques can be improved by using the concepts of logic programming. In their work, they have noticed certain constraints while translating DL axioms into a set of rules. A similar approach proposed by [Gómez et al., 2007] who show that a subset of DL languages can be effectively mapped into a set of strict and defeasible rules. Although we follow a similar approach proposed by [Gómez et al., 2007, Grosz et al., 2003], discussed in Chapter 6, while constructing a set of strict and defeasible rules from an ontology. Our purpose and application of those rules are quite different. We use those rules to build a context-aware system as multi-agent non-monotonic rule-based agents and use a distributed problem solving approach to see whether agents can infer certain contexts while they are resource-bounded.

3.10 Conclusion

The literature reviewed in this chapter is mainly focused on the reasoning formalisms for the semantic web. We have reviewed literatures on description logics, web ontology languages (OWL) and SWRL. Description logic has been considered as one of the most expressive formal languages having capability to perform reasoning about knowledge in an application domain. Description logic is based on ontology languages that are often used for context representation and reasoning. It is very suitable for modelling real life example systems. We have focused on ontology and SWRL because the work presented in this thesis is based on ontology-driven rule-based reasoning agents where rules are de-

rived from OWL 2 RL and SWRL. We also have discussed rule-based reasoning systems and multi-agent rule-based systems. The logical formalisms discussed in this thesis require rule-based reasoning technique to achieve the desired goals due to its simple and efficient reasoning mechanism. For non-monotonic reasoning, we have considered defeasible reasoning owing to its efficient reasoning capability; low computational complexity, and its focus on implementability. Defeasible reasoning is used to reason inconsistent and incomplete information. We have surveyed literature on monotonic as well as non-monotonic reasoning based logical formalisms including rule-based reasoning and the semantic web technologies. In doing so, we realized their efficacy towards context-aware systems. The ultimate purpose of considering this literature is to craft a comparative study that showcases the relevant and significant information regarding context-aware logical frameworks.

Chapter 4

The Logic \mathcal{L}_{OCRS}

4.1 Chapter Objectives

- To investigate a formal approach to context modelling and reasoning techniques.
- To develop a formal logical framework \mathcal{L}_{OCRS} for ontology-driven resource-bounded context-aware rule-based agents based on temporal epistemic description logic.
- To prove the correctness of \mathcal{L}_{OCRS} axiomatization such as soundness and completeness, validity and satisfiability and show how to express resource-bounded properties of the systems using \mathcal{L}_{OCRS} .

4.2 Motivation for the Logic \mathcal{L}_{OCRS}

The aim of this research is to develop a logical framework for modelling and reasoning about resource-bounded context-aware rule-based multi-agent systems. According to a survey [Rakib, 2012], various logical frameworks have been developed for modelling and verification of multi-agent systems. However, such frameworks may not be very suitable to model context-aware applications. This is because, most of those existing frameworks consider propositional logic as a simple knowledge representation language which is often not suitable for modelling real life complex systems. For example, propo-

sitional logic cannot directly talk about properties of individuals or relations between individuals. Much research in pervasive computing has been focused on incorporation of context-awareness features into pervasive applications by adapting the semantic web technology (see e.g., [Wang et al., 2004b, Esposito et al., 2008, Rakib and Faruqi, 2013]), where description logic (DL)-based ontology languages are often used for context representation and reasoning. DL is a decidable fragment of first order logic (FOL). In [Rakib and Faruqi, 2013], it has been shown how context-aware systems can be modelled as resource-bounded rule-based systems using ontologies. In that paper, the resources required by the agents to solve a given problem were considered the time and communication bandwidth, but not the space requirements for reasoning. Since context-aware systems often run on resource limited devices, memory requirements is an important factor for their reasoning.

In this chapter, we propose a logical framework based on the earlier work of Alechina et al. [Alechina et al., 2009a, Alechina et al., 2009c, Alechina et al., 2006], and the resulting logic \mathcal{L}_{OCRS} allows us to describe a set of ontology-driven rule-based reasoning agents with bound on time, memory, and communication. In addition to the incorporation of space (memory) requirements for reasoning in [Alechina et al., 2009a], \mathcal{L}_{OCRS} also uses first order Horn-clause rules derived from OWL 2 RL ontologies. While the frameworks presented in [Alechina et al., 2009a, Alechina et al., 2009c] provide a useful basis for experimentation with both the logical representation and verification of heterogeneous agents, it has become clear that a more expressive logical language is required if these frameworks are to be used for real world context-aware agents. Though the logic developed by [Alechina et al., 2006] is based on FOL, memory bounds have not been imposed in that framework. The proposed framework allows us to determine how much time (measured as rule-firing cycles) are required to generate certain contexts, how many messages must be exchanged between agents, and how much space (memory) is required for an agent for the reasoning. We provide an axiomatization of the logic and prove it is

sound and complete.

4.3 Formal Approaches to Resource-bounded Multi-agent System

As \mathcal{L}_{OCRS} is an extension of existing logics [Alechina et al., 2006, Alechina et al., 2009a, Alechina et al., 2009c], so it is worth discussing the relevant literature to show the novelty of the proposed \mathcal{L}_{OCRS} framework. In [Alechina et al., 2006], it is shown how to establish the correctness and time bounds for multi-agent systems considering distributed rule-based reasoning. The authors have emphasized on the beliefs of rule-based agents and shown how communicating agents adapt their behaviour over time using rule-based reasoning by developing a sound and complete modal logic. In that system, rule-based agents typically fire multiple rules to solve a particular problem. A simple case study has shown for specifying temporal properties in multi-agent systems using model checking techniques to verify properties of agents in a precise and realistic way. This system provides the quality of service guarantees, for example, each query is answered within n time steps (a step determines how a system moves from one state to the next considering a set of actions). However, memory and communication bound have not been considered in that framework.

In [Alechina et al., 2009a], a framework has presented for automated verification of time and communication requirements in the system of distributed rule based reasoning agents. This framework, \mathcal{L}_{CRB} , determines the number of rule-firing cycles; the number of messages exchanged among agents and the trade-offs between the communications and time in order to solve a particular problem. This framework has extended CTL^* with belief and communication modalities to represent the bounds on the communication by fixing the number of messages exchanged between agents. Each agent contains belief operators and communication modalities to provide sound and complete axiomatization of the

system. The logical model was verified using Mocha [Alur et al., 1998] model checker considering a typical example system. However, memory bounds has not been considered in that framework. Later in [Alechina et al., 2009c], Alechina et al. have proposed a logical framework based on temporal epistemic logic (*BMCL*) by incorporating memory bounds in addition to time and communication bounds. In that framework, distributed reasoners (agents) use resolution-based reasoning technique to solve a particular problem considering resource-bounds while the tradeoffs among these resources are carefully monitored. Although this framework has primarily considered the resource-bounds (time, memory and communication) in order to solve the problem, however primary focus is given on resolution-based reasoning technique with different set of actions and propositional inference rules.

In contrast to previous work discussed above, the extended framework presented in this chapter uses first order Horn clause rules derived from ontologies and allows rule-based reasoning considering time, memory and communication bounds in systems of context-aware reasoning agents.

4.4 Description Logic Based Reasoning

We have already discussed the essential concepts of description logics in Chapter 3. In this section we provide a very brief overview of description logic based reasoning to suitably implement the context modelling technique for context-aware rule-based reasoning agents. A *DL* knowledge base (*KB*) has two components: the Terminology Box (*TBox*) \mathcal{T} and the Assertion Box (*ABox*) \mathcal{A} . The *TBox* introduces the terminology of a domain, while the *ABox* contains assertions about individuals in terms of this vocabulary. The *TBox* is a finite set of general concept inclusions (*GCI*) and role inclusions. A *GCI* is of the form $C \sqsubseteq D$ where C, D are *DL*-concepts and a role inclusion is of the form $R \sqsubseteq S$ where R, S are *DL*-roles. We may use $C \equiv D$ (concept equivalence) as an abbreviation for the two *GCI*s $C \sqsubseteq D$ and $D \sqsubseteq C$ and $R \equiv S$ (role equivalence) as an abbreviation

for $R \sqsubseteq S$ and $S \sqsubseteq R$. The $ABox$ is a finite set of concept assertions in the form of $C(a)$ and role assertions in the form of $R(a, b)$, where a and b represent the individual names which are instances of the given role.

4.4.1 Context Modelling

In $\mathcal{L}_{O\mathcal{C}RS}$, we represent context as either $C(a)$ or $R(a, b)$. For context modelling we use OWL 2 RL, a profile of the new standardization OWL 2, which is based on the description logic program (DLP) [Grosz et al., 2003]. We choose OWL 2 RL for the design and development of rule-based systems because it is more expressive than the RDFS and suitable. Furthermore, we express more complex rule-based concepts using SWRL [Horrocks et al., 2004] which allow us to write rules using OWL concepts. In our framework, a context-aware system composed of a set of rule-based agents, and firing of rules that infer new facts may determine context changes and representing overall behaviour of the system.

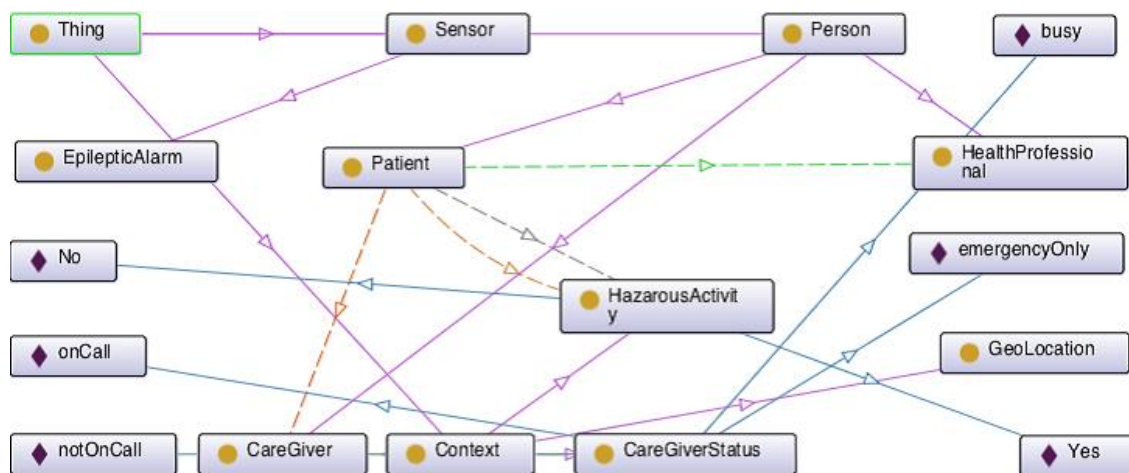


Figure 4.1: A Fragment of the Epileptic Patients' Monitoring Ontology

To illustrate the idea of how we model contexts using ontologies, we model here a health care domain considering an epilepsy scenario adapted from [Dockhorn Costa, 2007]. The scenario is based on the monitoring of epileptic patients to detect epileptic seizures. An epileptic alarm may activate several actions such as warning the patient about potential

Patient's rule
Initial facts: Patient('Tracy'), isAlarming('Tracy, 'Beep'), hasGeolocation('Tracy, 'DownTown') Patient(?p),isAlarming(?p,'s) \rightarrow EpilepticAlarm(?p) EpilepticAlarm(?p) \rightarrow hasHazardousActivity(?p, 'Yes) hasHazardousActivity(?p, 'Yes) \rightarrow isAgreed(?p,'Yes) hasHazardousActivity(?p, 'Yes) \rightarrow isAgreed(?p,'No) EpilepticAlarm(?p), hasHazardousActivity(?p, 'Yes), hasGeoLocation(?p, ?location) \rightarrow hasNotifiedPatient(?p, ?location) EpilepticAlarm(?p), hasHazardousActivity(?p, 'Yes), hasGeoLocation(?p, ?location), isAgreed(?p, 'Yes) \rightarrow Tell(1, 2, hasNotifiedPlanner(?p,?location))
Planner's rules
Initial facts: isCareGiverOf('Fiona,'Tracy), isCareGiverOnCall('Fiona, 'OnCall) Tell(1, 2, hasNotifiedPlanner(?p,?location)) \rightarrow hasNotifiedPlanner(?p,?location) hasNotifiedPlanner(?p,?location),lessThan(?location,'30), greaterThan(?location,0) \rightarrow situationWithinRange(?p,?location) situationWithinRange(?p,?location),isCareGiverOf(?c,'p),isCareGiverOnCall(?c,'stat) \rightarrow Ask(2,3, hasCareStatus(?c, 'stat)) Tell(3,2,hasCarStatus(?c,'onCall)) \rightarrow hasCarStatus(?c,'onCall) hasCarStatus(?c,'onCall), hasNotifiedPlanner(?p,?location) \rightarrow AcceptRequest(?c, ?p) Tell(3,2,hasCarStatus(?c, 'Busy)) \rightarrow hasCareStatus(?c, 'Busy) hasNotifiedPlanner(?p,?location) \rightarrow Tell(2, 5, hasNotifiedPlanner(?p,?location))
CareGiver's rules
Initial facts: Ask(2,3, hasCareStatus(?c, ?stat)) \rightarrow hasCareStatus(?c, ?stat) hasCareStatus(?c, ?stat) \rightarrow Tell(3,2,hasCarStatus(?c, 'OnCall)) hasCareStatus(?c, ?stat) \rightarrow Tell(3,2,hasCarStatus(?c, 'Busy)) hasCareStatus(?c, ?stat) \rightarrow Tell(3,2,hasCarStatus(?c, 'NotOnCall)) hasCareStatus(?c, ?stat) \rightarrow Tell(3,2,hasCarStatus(?c, 'EmergencyOnly))
HealthProfessional's rules
Initial facts: isHealthProfesional('John, 'Tracy) Tell(2, 5, hasNotifiedPlanner(?p,?location)) \rightarrow hasNotifiedPlanner(?p,?location) hasNotifiedPlanner(?p,?location), isHealthProfesional(?prof, ?p) \rightarrow logEpilepticAlarm(?prof,?p)

Table 4.1: Horn-Clause Rules for the Epileptic Patients' Monitoring Context-aware System

danger, informing patient's caregivers to take appropriate actions, and sending SMS messages to patient's relatives who are currently near to the patient, among others.

Rules
EpilepticAlarm(?p), hasHazardousActivity(?p, ?hazardousValue), isAgreed(?p, Yes), hasLocation(?p, ?loc) -> hasNotifiedPlanner(?p, ?loc)
hasCareStatus(?c, onCall), hasNotifiedPlanner(?p, ?loc) -> AcceptRequest(?c, ?p)
Patient(?p), isAlarming(?p, ?s) -> EpilepticAlarm(?p)
EpilepticAlarm(?p) -> hasHazardousActivity(?p, Yes)
hasNotifiedPlanner(?p, ?loc), greaterThan(?p, 0), lessThan(?loc, 100) -> situationWithinRange(?p, ?loc)
isHealthProfessionalOf(?prof, ?p), hasNotifiedPlanner(?p, ?loc) -> logEpilepticAlarm(?prof, ?p)

Figure 4.2: Example SWRL Rules

“The goal of the the epileptic patients' monitoring context-aware system is to detect the seizures, and to react in the following ways: (a) notify the epileptic patient of an upcoming seizure; and (b) notify his/her nearby caregivers of an upcoming seizure of the patient by showing a map with the location of the patient. The caregivers who receive

the notification for help should be (i) assigned as one of the caregivers of that particular patient; (ii) available for helping; and (iii) physically close to the patient. Upon a notification for help, caregivers may either accept or reject the request for helping the epileptic patient. When a particular caregiver accepts to help, the other caregivers who had received the notification for help are informed that a certain caregiver has already accepted to help that patient”[Dockhorn Costa, 2007].

Using Protégé [Protégé, 2011], we build an OWL 2 RL ontology to model the scenario and introduce the vocabulary, concepts and their relationships to represent the domain. This is to represent the static behavior of the system. A fragment of this ontology is depicted in Fig 4.1. The dynamic aspect of the system is captured using SWRL rules. A snapshot of some SWRL rules is given in Fig 4.2. In order to build context-aware systems as rule-based agents, we translate OWL 2 RL ontology into a set of Horn-clause rules. The combination of these translated rules with the user defined SWRL rules, which are already in the Horn-clause format, provide foundational knowledge to design the desired distributed rule-based agents. Protégé editor does not allow us to write communication rules with *Ask* and *Tell*. Therefore, these rules are written using annotations which are also translated into Horn-clause rules. There are four agents in this system: (1) Patient, (2) Planner, (3) CareGiver and (4) HealthProfesional. The set of translated rules with the initial facts are distributed to agents as shown in Table 4.1 for the epileptic patients’ monitoring context-aware system. Some interesting resource-bounded properties of the above system includes:

$$\begin{aligned} &G(B_1 \text{EpilepticAlarm}('Tracy)) \\ &\rightarrow X^n B_1 \text{Tell}(1, 2, \text{hasNotifiedPlanner}('Tracy, 'DownTown)) \\ &\quad \wedge (msg_1 =^m) \wedge (n_M(1) \geq l) \end{aligned}$$

the above property specifies that whenever there is an epileptic alarm for Tracy, agent 1 notifying agent 2 that "Tracy" has hazardous activity and she is located in "DownTown" within n time steps, while exchanging m messages and space requirement for agent 1 is at least l units, and

$$G(B_2 \text{ Tell}(1, 2, \text{hasNotifiedPlanner}(' \text{Tracy}', \text{DownTown})) \\ \rightarrow X^n B_2 \text{ AcceptRequest}(' \text{Fiona}', \text{Tracy}) \wedge (msg_2 =^m) \wedge (n_M(2) \geq l))$$

which specifies that whenever agent 2 gets notified that "Tracy" has hazardous activity and she is located in "DownTown" it believes that caregiver Fiona accepts the request within n time steps, while exchanging m messages and space requirement for agent 2 is at least l units.

4.5 \mathcal{L}_{OCS} - A Logic for Context-aware Systems

A multi-agent context-aware system consists of n_{A_g} agents, where $n_{A_g} \geq 1$. Each individual agent is identified by a value in $A_g = \{1, \dots, n_{A_g}\}$, and we use variables i and j over $\{1, \dots, n_{A_g}\}$ to refer to the agents. Each agent $i \in A_g$ has a program, consisting of ontology driven (OWL 2 RL and SWRL) Horn-clause rules and a working memory, which contains contexts (ground atomic facts) obtained from ABox knowledge base representing the initial state of the system. Rules are of the form of $P_1, P_2, P_3, \dots, P_n \rightarrow P$ where the antecedents $P_1, P_2, P_3, \dots, P_n$ and consequent P are context information. Antecedents of the rule are of the form of complex contexts (concepts and roles) which is a conjunction of n contexts. For each agent i , the rule instances are matched against the contents of the working memory to derive the consequent.

The resource-bounded context-aware system allows group of agents to cooperate with each other to share contextual information and infer new facts which no single agent could do alone. Thus sharing knowledge among agents is an efficient way of building context aware systems. Agents utilize a centralized common ontology and *Ask/Tell* communication mechanism to develop a distributed model. Using communication mechanism, agents exchange messages to derive contextual information. For this purpose, we need to model communication between agents. We assume that agents have two special communication primitives $Ask(i, j, P)$ and $Tell(i, j, P)$ in their language where i and j are agents and P is an atomic context not containing an *Ask* or a *Tell*. $Ask(i, j, P)$ stands for 'i asks j

about a context P ' and $Tell(i, j, P)$ means ' i tells j about a context P ' and $i \neq j$. The position in which these communication primitives may appear in a rule depends on which agent's program the rule belongs to. Communication rules may have an Ask or a $Tell$ with arguments (i, j, P) in the consequent, for example, $P_1, P_2, P_3, \dots, P_n \rightarrow Ask(i, j, P)$ [Alechina et al., 2006].

Figure 4.3 illustrates the exchange of information between two agents i and j . The exchange of information between agents works like this: if an $Ask(i, j, P)$ is in the working memory of agent i , $Ask(i, j, P)$ is not in the agent j 's working memory, and agent j has not exceeded its communication bound then $Ask(i, j, P)$ is added to the working memory of agent j and its communication counter is incremented. This action may also overwrite agent j 's memory due to memory bound (discussed in Section 4.5.3). We view the process of producing new contexts from existing contexts as a sequence of states of an agent, starting from an initial state. Similarly, when $Tell(j, i, P)$ is in the working memory of agent j , $Tell(j, i, P)$ is not in the agent i 's working memory, and agent i has not exceeded its communication bound then $Tell(j, i, P)$ is added to the working memory of agent i and its communication counter is incremented. Due to memory bound, an arbitrary context will be overwritten if agent's memory is full (discussed in Section 4.5.3).

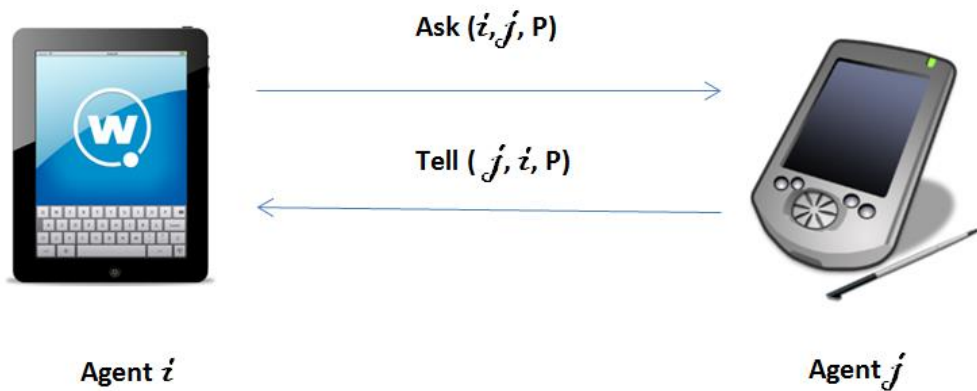


Figure 4.3: Two Agent's Communication

Apart from communication rules, all other rules are considered as deduction rules. A rule of the form $Tell(i, j, P) \rightarrow P$ is called trust rule. No other occurrences of Ask and $Tell$ are allowed.

The important point about OWL 2 is that it is limited to unary and binary predicates and it is function free. So using Protégé OWL editor, communication primitives are expressed by constant symbols. These annotated symbols are translated appropriately when designing the target system using Maude specification for system verification (discussed in Chapter 6).

4.5.1 The Language of \mathcal{L}_{OCRS}

Now we define the internal language of \mathcal{L}_{OCRS} of each agent in the system. This logic is an extension of the logic developed by [Alechina et al., 2009a]. To specify contextual information more precisely, let $\mathcal{C} = \{C_1, C_2, \dots, C_n\}$ be a finite set of concept names, $\mathcal{R} = \{R_1, R_2, \dots, R_n\}$ be the finite set of role names and \mathcal{A} be the finite set of assertions. For communication primitives, we define a set $\mathcal{Q} = \{Ask(i, j, P), Tell(i, j, P)\}$, where $i, j \in A_g$ and $P \in \mathcal{C} \cup \mathcal{R}$. Note that \mathcal{C} and \mathcal{R} are the concept and role names that appear in \mathcal{A} .

Let $\mathfrak{R} = \{r_1, r_2, \dots, r_n\}$ be a finite set of rules of the form $P_1, P_2, \dots, P_n \rightarrow P$, where $n \geq 0$, $P_i, P \in \mathcal{C} \cup \mathcal{R} \cup \mathcal{Q}$ for all $i \in \{1, 2, \dots, n\}$ and $P_i \neq P_j$ for all $i \neq j$. For convenience, we use the notation $ant(r)$ for the set of antecedents of r and $cons(r)$ for the consequent of r , where $r \in \mathfrak{R}$. Let $g : \wp(\mathcal{A}) \rightarrow \mathfrak{R}$ be a substitution function that uses a forward-chaining strategy to instantiate the rule-base. We denote by $\mathcal{G}(\mathfrak{R})$ the set of all the ground instances of the rules occurring in \mathfrak{R} , which is obtained using g . Thus $\mathcal{G}(\mathfrak{R})$ is finite. Let $\bar{r} \in \mathcal{G}(\mathfrak{R})$ be one of the possible instances of a rule $r \in \mathfrak{R}$.

Note that $C(a)$, $R(a, b)$, $Ask(i, j, C(a))$, $Ask(i, j, R(a, b))$, $Tell(i, j, C(a))$, and $Tell(i, j, R(a, b))$ are ground facts, for all $C \in \mathcal{C}$, $R \in \mathcal{R}$. The internal language \mathcal{L}

includes all the ground facts and rules. Let us denote the set of all formulas by Ω which is finite. In the modal language of \mathcal{L} we have belief operator B_i for all $i \in A_g$.

4.5.2 Communication Bound

We assume that there is a bound on communication for each agent i which limits agent i to at most $n_C(i) \in \mathbb{Z}^*$ messages. Each agent has a communication counter $cp_i^{\bar{n}}$ to keep record of agent's communication, which starts from 0 ($cp_i^{\bar{0}}$) and is not allowed to exceed the value $n_C(i)$. The value of communication counter increases for every single communication for each agent.

For the communication bound, we define the following set:

$$CP_i = \{cp_i^{\bar{n}} | n = \{0, \dots, n_C(i)\}\},$$

$$CP = \bigcup_{i \in A_g} CP_i.$$

4.5.3 Memory Bound

We divide agent's memory into two parts as rule memory (knowledge base) and working memory. Rule memory holds the set of rules, whereas the facts are stored in the agent's working memory. Working memory of an agent i is divided into static memory ($S_M(i)$) and dynamic memory ($D_M(i)$). The static part $S_M(i)$ contains initial information to start up the systems, for example, initial working memory facts. Thus its size is determined by the number of initial facts. The dynamic part $D_M(i)$ of each agent $i \in A_g$ is bounded in size by $n_M(i) \in \mathbb{Z}^*$, where one unit of memory corresponds to the ability to store an arbitrary context. The dynamic part $D_M(i)$ contains newly derived facts as the system moves. Only formulas stored in $D_M(i)$ may get overwritten if it is full. The question arises here why a context is overwritten arbitrarily and not in a sequential manner? As the static segment of the memory contains the set of initial contexts (facts) to derive the rules, so a context can be arbitrarily overwritten in the dynamic segment of the memory which enhances the system's efficiency. Apart from this, the system has concurrent transition

system by firing the set of rule instances, so it is very hard to figure out and overwrite a context which has the least priority to be used. Note that unless otherwise stated, in the rest of the thesis we shall assume that memory means $D_M(i)$.

4.5.4 Syntax

The syntax of $\mathcal{L}_{O\mathcal{CRS}}$ includes temporal operators of CTL^* and is defined inductively as follows:

- \top (tautology) and *start* (a propositional variable which is only true at the initial moment of time) are well-formed formulas (wff) of $\mathcal{L}_{O\mathcal{CRS}}$;
- cp_i^n (which states that the value of agent i 's communication counter is n) is a wff of $\mathcal{L}_{O\mathcal{CRS}}$ for all $n \in \{0, \dots, n_C(i)\}$ and $i \in A_g$;
- $B_i C(a)$ (agent i believes $C(a)$), $B_i R(a, b)$ (agent i believes $R(a, b)$), and $B_i r$ (agent i believes r) are wffs of $\mathcal{L}_{O\mathcal{CRS}}$ for any $C \in \mathcal{C}$, $R \in \mathcal{R}$, $r \in \mathfrak{R}$ and $i \in A_g$;
- $B_k Ask(i, j, C(a))$, $B_k Ask(i, j, R(a, b))$, $B_k Tell(i, j, C(a))$, and $B_k Tell(i, j, R(a, b))$ are wffs of $\mathcal{L}_{O\mathcal{CRS}}$ for any $C \in \mathcal{C}$, $R \in \mathcal{R}$, $i, j \in A_g$, $k \in \{i, j\}$, and $i \neq j$;
- If φ and ψ are wffs of $\mathcal{L}_{O\mathcal{CRS}}$, then so are $\neg\varphi$ and $\varphi \wedge \psi$;
- If φ and ψ are wffs of $\mathcal{L}_{O\mathcal{CRS}}$, then so are $X\varphi$ (in the next state φ), $\varphi U\psi$ (φ holds until ψ), $A\varphi$ (on all paths φ).

Other classical abbreviations for \perp , \vee , \rightarrow and \leftrightarrow , and temporal operations are defined as usual which are given as:

- $F\varphi \equiv \top U\varphi$; $F\varphi$ means that φ will appear at some point in the future and $\top U\varphi$ states that true appears in all states until φ appears on the state. Both are equivalent.
- $G\varphi \equiv \neg F\neg\varphi$; where $G\varphi$ states that φ appears globally or in other words, $\neg\varphi$ will never appear in future states.

- $E\varphi \equiv \neg A\neg\varphi$ where $E\varphi$ means φ appears on some paths. In other words, we can say that $\neg\varphi$ will never appear on all paths.

4.5.5 Semantics

The semantics of $\mathcal{L}_{O\mathcal{C}\mathcal{R}\mathcal{S}}$ is defined by $\mathcal{L}_{O\mathcal{C}\mathcal{R}\mathcal{S}}$ transition system. The structure of the transitions, to and from the states, is in ω -tree shape. The states corresponds to the working memory and record of communication counter. In the formal models of multi-agent systems, agents fire multiple rule instances to achieve the desired goal state. Each agent may derive new contexts whenever it has matching rules regardless of looking into other agents' activities whether they are firing rules or idle in the system. When an agent derives a new context by firing a rule instance, then the system moves to the next state. In other words, we can say that a transition corresponds to exactly one inferred context (by firing a rule instance) or a transition corresponds to copying one context when communicating with another agent. However in case, if an agent has no rule instance to fire or does not communicate with any other agent then the system transits to next state with the same set of contexts using idle rule.

Let (S, T) be a pair where S is a set of states and T is a binary relation on S that is total, i.e., $\forall s \in S, \exists s' \in S$ such that sTs' where s is the current state and s' is the successor state of the transition. The $\mathcal{L}_{O\mathcal{C}\mathcal{R}\mathcal{S}}$ transition system is based on ω -tree structures, which contains finite number of branches. A branch of (S, T) is an ω -sequence (s_0, s_1, \dots) such that s_0 is the root and s_iTs_{i+1} for all $i \geq 0$. We denote $B(S, T)$ to be the set of all branches of (S, T) . For a branch $\pi \in B(S, T)$, π_i denotes the element s_i of π and $\pi_{\leq i}$ is the prefix (s_0, s_1, \dots, s_i) of π . A $\mathcal{L}_{O\mathcal{C}\mathcal{R}\mathcal{S}}$ transition system \mathbb{M} is defined as $\mathbb{M} = (S, T, V)$ where

- (S, T) is a ω -tree frame
- $V : S \times A_g \rightarrow \wp(\Omega \cup CP)$; we define the belief part of the assignment $V^B(s, i) = V(s, i) \setminus CP$ and the communication counter part $V^C(s, i) = V(s, i) \cap CP$.

We further define $V^M(s, i) = \{\alpha | \alpha \in D_M(i)\}$.

V satisfies the following conditions:

1. $|V^c(s, i)| = 1$ for all $s \in S$ and $i \in A_g$.
 2. If sTs' and $cp_i^{\bar{n}} \in V(s, i)$ and $cp_i^{\bar{m}} \in V(s', i)$ then $n \leq m$.
- We say that a rule $r : P_1, P_2, \dots, P_n \rightarrow P$ is applicable in a state s of an agent i if $ant(\bar{r}) \in V(s, i)$ and $cons(\bar{r}) \notin V(s, i)$. The following conditions on the assignments $V(s, i)$, for all $i \in A_g$, and transition relation T holds in all models:
 1. For all $i \in A_g$, $s, s' \in S$, and $r \in \mathfrak{R}$, $r \in V(s, i)$ iff $r \in V(s', i)$. This describes that agent's program does not change, i.e., rules are neither added nor deleted during execution.
 2. For all $s, s' \in S$, sTs' holds iff for all $i \in A_g$, $V(s', i) = V(s, i) \cup \{cons(\bar{r})\} \cup \{Ask(j, i, C(a))\} \cup \{Tell(j, i, C(a))\} \cup \{Ask(j, i, R(a, b))\} \cup \{Tell(j, i, R(a, b))\}$. This describes that each agent i fires a single applicable rule instance of a rule r , or updates its state by interacting with other agents, otherwise its state does not change.

The truth of a $\mathcal{L}_{O CRS}$ formula at a point n of a path $\pi \in B(S, T)$ is defined inductively as follows:

- $\mathbb{M}, \pi, n \models \top$
- $\mathbb{M}, \pi, n \models start$ iff $n = 0$,
- $\mathbb{M}, \pi, n \models B_i \alpha$ iff $\alpha \in V(\pi_n, i)$.
- $\mathbb{M}, \pi, n \models cp_i^{\bar{m}}$ iff $cp_i^{\bar{m}} \in V(\pi_n, i)$.
- $\mathbb{M}, \pi, n \models \neg \varphi$ iff $\mathbb{M}, \pi, n \not\models \varphi$.
- $\mathbb{M}, \pi, n \models \varphi \vee \psi$ iff $\mathbb{M}, \pi, n \models \varphi$ or $\mathbb{M}, \pi, n \models \psi$.

- $\mathbb{M}, \pi, n \models \varphi \wedge \psi$ iff $\mathbb{M}, \pi, n \models \varphi$ and $\mathbb{M}, \pi, n \models \psi$.
- $\mathbb{M}, \pi, n \models X\varphi$ iff $\mathbb{M}, \pi, n+1 \models \varphi$.
- $\mathbb{M}, \pi, n \models \varphi U \psi$ iff $\exists m \geq n$ such that $\forall k \in [n, m)$ $\mathbb{M}, \pi, k \models \varphi$ and $\mathbb{M}, \pi, m \models \psi$
- $\mathbb{M}, \pi, n \models A\varphi$ iff $\forall \pi' \in B(S, T)$ such that $\pi'_{\leq n} = \pi_{\leq n}$ $\mathbb{M}, \pi', n \models \varphi$

We now describe conditions on the models. The transition relation T corresponds to the agent's executing actions $\langle act_1, act_2, \dots, act_{n_{Ag}} \rangle$, where act_i is a possible action of an agent i in a given state s . The set of actions that each agent i can perform are as follows:

- *Rule_{i,r, β}* : An agent i fires a matching rule instance \bar{r} and adds consequent $cons(\bar{r})$ to its working memory and remove arbitrary context β in case if the memory is full. If memory of agent i is not full then newly derived context will be placed into an empty cell of its memory.
- *Copy_{i, α , β}* : An agent i copies a context α from other agent j 's working memory and removes β from its own working memory if the memory is full, where α is of the form $Ask(j, i, P)$ or $Tell(j, i, P)$. *Copy* action is triggered by firing communication rule instances.
- *Idle_i*: For idle action, agent i moves to the next states and leaves its configuration unchanged.

We denote arbitrary context by β which gets overwritten if it is in the agent's dynamic memory ($D_M(i)$). If agent's memory is full $|V(s, i)| = n_M(i)$ then we require that β has to be in $V^M(s, i)$. All actions are not possible to be performed in any given state, e.g.; there may not be any matching rule instances. When the counter value reaches to $n_C(i)$, then i cannot perform further copy actions. Let us denote the set of all possible actions by agent i in a given state s by $T_i(s)$ and its definition is given below:

Definition 4.1. Available Actions

An agent i moves from state s to s' by performing act_i . The set of actions that each agent i can perform are:

1. $Rule_{i,r,\beta} \in T_i(s)$ iff $r \in V(s, i)$, $ant(\bar{r}) \subseteq V(s, i)$, $cons(\bar{r}) \notin V(s, i)$, $\beta \in \Omega$ or if $|V^M(s, i)| = n_M(i)$ then $\beta \in V^M(s, i)$;

Agent i firing a rule instance \bar{r} and adding $cons(\bar{r})$ to its working memory and removing β ,

2. $Copy_{i,\alpha,\beta} \in T_i(s)$ iff there exists $j \neq i$ such that $\alpha \in V(s, j)$, $\alpha \notin V(s, i)$, $cp_i^{\bar{n}} \in V(s, i)$ for some $n < n_C(i)$, α is of the form $Ask(j, i, P)$ or $Tell(j, i, P)$, and β as before;

Agent i copying α from other agent's memory and removing β , where α is either $Ask(j, i, P)$ or $Tell(j, i, P)$ and its communication counter has not exceeded the communication bound.

3. $Idle_i$ is always in $T_i(s)$.

Agent i does nothing but moves to the next state.

Definition 4.2. Effects of Action

For each $i \in A_g$, the result of performing an action act_i in state $s \in S$ is defined if $act_i \in T_i(s)$ and has the following effect on the assignment of formulas to i in the successor state $s' \in S$:

1. if act_i is $Rule_{i,r,\beta}$: $V(s', i) = V(s, i) \setminus \{\beta\} \cup \{cons(\bar{r})\}$;
2. if act_i is $Copy_{i,\alpha,\beta}$: $cp_i^{\bar{m}} \in V(s, i)$ for some $m < n_C(i)$: $V(s', i) = V(s, i) \setminus \{\beta, cp_i^{\bar{m}}\} \cup \{\alpha, cp_i^{\bar{m}+1}\}$;
3. if act_i is $Idle_i$: $V(s', i) = V(s, i)$.

Now, the definition of the set of models corresponding to a system of rule-based reasoners is given below:

Definition 4.3. $\mathbb{M}(n_M, n_C)$ is the set of models (S, T, V) which satisfies the following conditions:

1. $cp_i^{\bar{0}} \in V(s_0, i)$ where $s_0 \in S$ is the root of (S, T) for all $i \in A_g$;
2. $\forall s \in S$ and a tuple of actions $\langle act_1, act_2, \dots, act_{n_{Ag}} \rangle$, if $act_i \in T_i(s), \forall i \in A_g$, then $\exists s' \in S$ such that sTs' and s' satisfies the effects of $act_i, \forall i \in A_g$;
3. $\forall s, s' \in S$ and sTs' iff for some tuple of actions $\langle act_1, act_2, \dots, act_{n_{Ag}} \rangle, act_i \in T_i(s)$ and the assignment in s' satisfies the effects of $act_i, \forall i \in A_g$;
4. The bound on each agent's memory and communication is set by the following constraint on the mapping $V : |V^M(s, i)| \leq n_M(i)$, and $cp_i^{\bar{n}} \leq n_C(i)$ for all $s \in S$, and $i \in A_g$.

The definition 4.3 describes the set of conditions for the class $\mathbb{M}(n_M, n_C)$ to model context-aware rule-based reasoning agents using \mathcal{L}_{OCRS} transition system. The first condition sets the communication counter value to 0 at the initial state s_0 of agent i . Note that the bound $n_C(i)$ on each agent i 's communication ability (no branch contains more than $n_C(i)$ *Copy* actions by agent i) follows from the fact that $Copy_i$ is only enabled if i has performed fewer than $n_C(i)$ copy actions in the past. Second and third conditions describe the transition relation according to their corresponding actions. The last condition sets the memory and communication bounds for all agents in the system. Below are some abbreviations which will be used in the axiomatization:

- $ByRule_i(P, m) = \neg B_i P \wedge cp_i^{\bar{m}} \wedge \bigvee_{r \in \mathcal{R} \wedge cons(\bar{r})=P} (B_i r \wedge \bigwedge_{Q \in ant(\bar{r})} B_i Q)$.

This formula describes the state before the agent comes to believe formula P by the *Rule* transition, m is the value of i 's communication counter, P and Q are ground atomic formulas.

- $ByCopy_i(\alpha, m) = \neg B_i \alpha \wedge B_j \alpha \wedge cp_i^{\bar{m}-1}$, where α is of the form $Ask(j, i, P)$ or $Tell(j, i, P), i, j \in A_g$ and $i \neq j$.

This formula describes the state before the agent comes to believe formula α by the *Copy* action (by firing communication rule instance), m is the value of i 's communication counter, and α is the formula copied from other agent's memory.

4.5.6 Axiomatization

Now we introduce the axiomatization system.

A1. All axioms and inference rules of CTL* [Reynolds, 2001].

$$A2 \quad \bigwedge_{\substack{\alpha \in D_M(i) \\ D_M(i)}} B_i \alpha \rightarrow \neg B_i \beta \text{ for all } D_M(i) \subseteq \Omega \text{ such that } |D_M(i)| = n_M(i) \text{ and } \beta \notin D_M(i).$$

This axiom describes that, in a given state, each agent can store maximally at most $n_M(i)$ formulas in its memory,

$$A3. \quad \bigvee_{n=0, \dots, n_C(i)} cp_i^{\bar{n}}.$$

This axiom says the value of the communication counter in the agent i 's memory is n which should be between 0 and $n_C(i)$. This corresponds to the *Copy* action performed by agent i .

$$A4. \quad cp_i^{\bar{n}} \rightarrow \neg cp_i^{\bar{m}} \text{ for any } m \neq n.$$

The value of the communication counter n must be unique. It may change due to *Copy* action, otherwise its value remains the same as in the previous state.

$$A5 \quad B_i r \wedge \bigwedge_{P \in ant(\bar{r})} B_i P \wedge cp_i^{\bar{n}} \wedge \neg B_i cons(\bar{r}) \rightarrow EX(B_i cons(\bar{r}) \wedge cp_i^{\bar{n}}), i \in A_g.$$

This axiom describes that a transition is always possible in a state whenever agent i believes on the consequent of the rule $cons(\bar{r})$ in its working memory and agent i 's communication counter does not change.

$$A6 \quad cp_i^{\bar{m}} \wedge \neg B_i \alpha \wedge B_j \alpha \rightarrow EX(B_i \alpha \wedge cp_i^{\bar{m}+1}) \text{ where } \alpha \text{ is of the form } Ask(j, i, P) \text{ or } Tell(j, i, P), i, j \in A_g, j \neq i, m < n_C(i).$$

This axiom describes transitions made by *Copy* with communication counter increased. So, the successor state extends their predecessors by adding one new belief which include a newly copied context with communication counter value incremented by 1.

A7 $EX(B_i\alpha \wedge B_i\beta) \rightarrow B_i\alpha \vee B_i\beta$, where α and β are not of the form $Ask(j, i, P)$ and $Tell(j, i, P)$.

This axiom says that at most one new belief is added in the next state and $\alpha \neq \beta$. So, either α or β appear in the memory.

A8 $B_i\alpha \rightarrow AX B_i\alpha$ for any $\alpha \in S_M(i) \cup \mathfrak{R}$.

This axiom states that an agent $i \in A_g$ always believes formulas residing in its static memory and its rules.

A9 $EX(B_i\alpha \wedge cp_i^m) \rightarrow B_i\alpha \vee ByRule_i(\alpha, m) \vee ByCopy_i(\alpha, m)$ for any $\alpha \in \cup \Omega$.

This axiom says that a new belief can only be added by one of the valid reasoning actions.

A10a. $start \rightarrow cp_i^0$, for all $i \in A_g$.

At the start state, the agent has not performed any *Copy* actions.

A10b. $\neg EX start$.

This axiom states that a propositional variable *start* holds only at the root of the tree.

A11. $B_i r$ where $r \in \mathfrak{R}$ for all $i \in A_g$.

This axiom tells agent i believes its own set of rules.

A12. $\neg B_i r$ where $r \notin \mathfrak{R}$ for all $i \in A_g$.

This axiom tells agent i only believes its rules.

A13 $\varphi \rightarrow EX\varphi$, where φ does not contain *start*.

This axiom describes an *Idle* transition by all the agents.

A14 $\bigwedge_{i \in A_g} EX(\bigwedge_{\alpha \in \Gamma_i} B_i \alpha \wedge cp_i^{\overline{m_i}}) \rightarrow EX \bigwedge_{i \in A_g} (\bigwedge_{\alpha \in \Gamma_i} B_i \alpha \wedge cp_i^{\overline{m_i}})$ for any $\Gamma_i \subseteq \Omega$.

This axiom describes that if each agent i can separately reach a state where it believes formulas in Γ_i , then all agents together can reach a state where for each i , agent i believes formulas in Γ_i .

Let us now define the logic obtained from the above axiomatisation system.

Definition 4.4. $\mathbb{L}(n_M, n_C)$ is the logic defined by the axiomatisation A1 – A14.

Theorem 4.1. $\mathbb{L}(n_M, n_C)$ is sound and complete with respect to $\mathbb{M}(n_M, n_C)$.

4.6 Soundness Proof

The proof of soundness is standard. In this section, soundness of the $\mathbb{L}(n_M, n_C)$ is proved by showing the validity of all axioms in $\mathbb{M}(n_M, n_C)$. The \mathcal{L}_{OCRS} actions preserve the validity of all axioms. For any given \mathcal{L} -formula φ and the set of formulas Ω , we can say that if $\Omega \vdash_{\mathbb{M}} \varphi$ then $\Omega \models_{\mathbb{M}} \varphi$. These proofs are given below:

- The proofs for axioms and rules included in A1 are given in [Reynolds, 2001].
- Axiom A2 assures that at a state, each agent can store maximally at most $n_M(i)$ formulas in its memory.

Let $\mathbb{M} = (S, T, V) \in \mathbb{M}(n_M, n_C)$, $\pi \in B(S, T)$ and $n \geq 0$. We assume $\mathbb{M}, \pi, n \models \bigwedge_{\alpha \in D_M(i)} B_i \alpha$, where $\alpha \in V(\pi_n, i)$ and $\beta \notin V(\pi_n, i)$, for any $\pi_n (= s) \in S$, and $|D_M(i)| = n_M(i)$. As the memory is full at this moment, so the current state of agent i does not allow any additional formula β to be added. The formula β can only be replaced with α if one of the applicable actions either $Rule_{i,r,\beta}$ or $Copy_{i,\alpha,\beta}$ is performed. According to the definition of $\mathbb{M}(n_M, n_C)$, $\exists s' \in S$ such that π_n

$T s'$ and $V(s', i) = V(\pi_n, i) \setminus \{\alpha\} \cup \{\beta\}$. Therefore, it is obvious then that

$$\mathbb{M}, \pi, n \models \bigwedge_{\alpha \in D_M(i)} B_i \alpha \rightarrow \neg B_i \beta.$$

- Axioms A3 and A4 force the existence of a unique communication counter for each agent to record the number of copies it has performed so far. More specifically, A3 assures that at least one counter value exists for any agent and A4 guaranties that only one of them is present.

For the given communication counter formula $cp_i^{\bar{n}}$, we assume $\mathbb{M}, \pi, n \models cp_i^{\bar{n}}$, where $cp_i^{\bar{n}} \in V(\pi_n, i)$, then $\mathbb{M}, \pi, n \not\models cp_i^{\bar{m}}$ for any $n \neq m$. Agent i can not have duplicate copies of communication counter.

- In the following, we provide the proof for A5.

Let $\mathbb{M} = (S, T, V) \in \mathbb{M}(n_M, n_C)$, $\pi \in B(S, T)$ and $n \geq 0$. We assume that $\mathbb{M}, \pi, n \models B_i r \wedge \bigwedge_{P \in \text{ant}(\bar{r})} B_i P \wedge cp_i^{\bar{m}} \wedge \neg B_i \text{cons}(\bar{r})$, for some $r \in \mathfrak{R}$ and $|V^M(s, i)| \leq n_M(i)$. Then $P \in V(\pi_n, i)$ for all $P \in \text{ant}(\bar{r})$, and $\text{cons}(\bar{r}) \notin V(\pi_n, i)$. This means that the action performed by agent i is $\text{Rule}_{i,r,\beta}$. According to the definition of $\mathbb{M}(n_M, n_C)$, $\exists s' \in S$ such that $\pi_n T s'$ and $V(s', i) = V(\pi_n, i) \setminus \{\beta\} \cup \{\text{cons}(\bar{r})\}$. Let π' be a branch in $B(S, T)$ such that $\pi'_{\leq n} = \pi_{\leq n}$ and $\pi'_{n+1} = s'$. Then, we have $\mathbb{M}, \pi', n+1 \models B_i \text{cons}(\bar{r}) \wedge cp_i^{\bar{m}}$. Therefore, it is obvious then that $\mathbb{M}, \pi, n \models EX(B_i \text{cons}(\bar{r}) \wedge cp_i^{\bar{m}})$.

- Axiom A6 is valid by copy action.

Let $\mathbb{M} = (S, T, V) \in \mathbb{M}(n_M, n_C)$, $\pi \in B(S, T)$ and $n \geq 0$. We assume that $\mathbb{M}, \pi, n \models cp_i^{\bar{m}} \wedge \neg B_i \alpha \wedge B_j \alpha$, and $|V^M(s, i)| \leq n_M(i)$. Then $cp_i^{\bar{m}} \in V(\pi_n, i)$, $\alpha \notin V(\pi_n, i)$, and $\alpha \in V(\pi_n, j)$ for $i, j \in A_g$, $i \neq j$, and $m < n_C(i)$. This means that the action performed by agent i is $\text{Copy}_{i,\alpha,\beta}$. According to the definition of $\mathbb{M}(n_M, n_C)$, $\exists s' \in S$ such that $\pi_n T s'$ and $V(s', i) = V(\pi_n, i) \setminus \{\beta, cp_i^{\bar{m}}\} \cup \{\alpha, cp_i^{\bar{m}+1}\}$. Let π' be a branch in $B(S, T)$ such that $\pi'_{\leq n} = \pi_{\leq n}$ and $\pi'_{n+1} = s'$. Then we have $\mathbb{M}, \pi', n+1 \models B_i \alpha \wedge cp_i^{\bar{m}+1}$. Therefore, it is obvious that $\mathbb{M}, \pi, n \models EX(B_i \alpha \wedge cp_i^{\bar{m}+1})$.

$$cp_i^{=m+1}).$$

- In A7: At most one new formula is added to some future states, i.e., $V(s', i) = V(s, i) \cup \{\alpha\} \cup \{\beta\}$. For the proof, we assume that $\mathbb{M}, \pi, n \models EX(B_i\alpha \wedge B_i\beta)$ where $\alpha, \beta \in \Omega$ and $\alpha, \beta \notin V(\pi_n, i)$ for any $i \in A_g$. This means that the action performed by agent i is $Rule_{i,r,\gamma}$. According to the definition of $\mathbb{M}(n_M, n_C)$, there exists a $s' \in S$ such that $\pi_n T s'$ and $V(s', i) = V(\pi_n, i) \setminus \gamma \cup \{\alpha\} \cup \{\beta\}$ where γ is an arbitrary context. Let π' be a branch in $B(S, T)$ such that $\pi'_{\leq n} = \pi_{\leq n}$ and $\pi'_{n+1} = s'$. Then we have either $\mathbb{M}, \pi, n+1 \models B_i\alpha$ or $\mathbb{M}, \pi, n+1 \models B_i\beta$.
- Axiom A8 states that the formula α is always valid for any agent $i \in A_g$. For this axiom, α is considered as an initial fact which is permanently stored in the static memory. So, an agent $i \in A_g$ always believes the formula $B_i\alpha$ in the static memory segment of all future states. The formulas in the static memory do not change during the execution of the system and the same for its rules.
- Axiom A9 is valid by one of the valid reasoning actions. For the given model $\mathbb{M} = (S, T, V) \in \mathbb{M}(n_M, n_C)$, we assume that $\mathbb{M}, \pi, n \models EX(B_i\alpha \wedge cp_i^{=m})$, for any $\alpha \in \Omega$ and $cp_i^{=m} \in V(\pi_n, i)$ where $|V^M(s, i)| \leq n_M(i)$ and for all $i \in A_g$. This means a new belief is added by one of the valid reasoning actions performed by agent i . According to the definition of $\mathbb{M}(n_M, n_C)$, there exists a $s' \in S$ such that $\pi_n T s'$ and $V(s', i) = V(\pi_n, i) \setminus \beta \cup \{\alpha\} \cup ByRule_i(\alpha, m) \cup ByCopy_i(\alpha, m)$.
- A10a is valid since the *start* appears at the root state of each agent $i \in A_g$ and the value of the communication counter is 0, $cp_i^{=0}$ because no copy action is performed.
- A10b is also valid because *start* only appears at the root of the tree.
- A11 is valid and states that agent i believes on the rule at state $s : V(\pi_n, i) = B_ir$ where $r \in \mathfrak{R}$ and rules are neither added nor deleted during the execution of the system.

- A12 is valid and states that agent i believes only its own rule at state s : $V(\pi_n, i) = \neg B_i r$ where $r \notin \mathcal{R}$. In our model, agents do not exchange rules.
- Axiom A13 is valid for the action *Idle*. Let us say if a formula $\varphi \in V(\pi_n, i)$, then there exists $s' \in S$ such that $\pi_n T s'$ and $V(s', i) = V(\pi_n, i)$.
- Axiom A14 is valid, if the set of actions performed by agent i to reach the goal state where agent i believes formulas in Γ_i , then all agents can simultaneously reach the goal state where for each agent i , agent i believes formulas in Γ_i .

4.7 Completeness Proof

Now we demonstrate the completeness proof of the axiomatization. Completeness can be shown by constructing a tree model for a consistent formula φ . This is constructed as in the completeness proof introduced in [Reynolds, 2001]. Then we use the axioms to show that this model is in $\mathbb{M}(n_M, n_C)$.

Since the initial state of all agents does not restrict the set of formulas they may derive in the future, for simplicity we conjunctively add to φ a tautology that contains all the potentially necessary formulas and message counters, in order to have enough sub-formulas for the construction. We construct a model $\mathbb{M} = (S, T, V)$ for

$$\varphi' = \varphi \bigwedge_{\alpha \in \Omega} (XB_i \alpha \vee \neg XB_i \alpha) \wedge \bigwedge_{n \in \{0, \dots, n_C(i)\}, i \in A_g} (Xcp_i^{\overline{n}} \vee \neg Xcp_i^{\overline{n}})$$

We then prove that \mathbb{M} is in $\mathbb{M}(n_M, n_C)$ such that it satisfies all those properties mentioned in definition 4.3.

- Axiom A2 assures that $|V^M(s, i)| \leq n_M(i)$ and each agent i can maximally store at most $n_M(i)$ formulas, i.e., $\alpha \in D_M(i)$ and $\beta \notin D_M(i)$. We need to prove that for all $s \in S$, $act_i \in T_i(s)$ and $i \in A_g$, there exists a $s' \in S$ such that $s T s'$ and $V(s', i)$ is the resultant state of $V(s, i)$ after i has performed action act_i . Let us consider the case when act_i is either $Rule_{i,r,\beta}$ or $Copy_{i,\alpha,\beta}$. As $D_M(i) = n_M(i)$, when action

act_i is performed, then one of the α is replaced. Therefore, $V(s', i) = V(s, i) \setminus \{\beta\} \cup \{\alpha\}$.

- Axiom A3 shows the presence of the communication counter value $cp_i^{\bar{m}} \in V(s, i)$ where $m \in \{0, \dots, n_C(i)\}$ for any $i \in A_g$.
- Axiom A4 assures the existence of a unique value at a state s of \mathbb{M} , $cp_i^{\bar{m}} \in V(s, i)$.
- For the proof of axiom 5, we need to prove that for all $s \in S$, $act_i \in T_i(s)$ and $i \in A_g$, there exists a $s' \in S$ such that $s T s'$ and $V(s', i)$ is the resultant state of $V(s, i)$ after i has performed action act_i . Let us consider the case when act_i is $Rule_{i,r,\beta} \in T_i(s)$ for some $r \in \mathfrak{R}$. Since the rule $Rule_{i,r,\beta}$ is applicable at s , $ant(\bar{r}) \subseteq V(s, i)$ and $cons(\bar{r}) \notin V(s, i)$. Therefore, there exist a MCS¹ (Maximal Consistent Set) χ such that $\chi \supseteq V(s, i)$, and $\bigwedge_{P \in ant(\bar{r})} B_i P \wedge cp_i^{\bar{m}} \wedge \neg B_i cons(\bar{r}) \in \chi$, for some $m \in \{0, \dots, n_C(i)\}$ and $|V^M(s, i)| \leq n_M(i)$. By Axiom A5 and MP (Modus Ponens), $EX(B_i cons(\bar{r}) \wedge cp_i^{\bar{m}}) \in \chi$. Therefore, according to the construction, $\exists s' \in S$ such that $s T s'$, $V(s', i) \subseteq \chi'$ for some χ' , and $B_i cons(\bar{r}) \wedge cp_i^{\bar{m}} \in \chi'$. Therefore, $V(s', i) = V(s, i) \setminus \{\beta\} \cup \{cons(\bar{r})\}$.
- For the proof of axiom A6, we need to prove that for all $s \in S$, $act_i \in T_i(s)$ and $i \in A_g$, there exists a $s' \in S$ such that $s T s'$ and $V(s', i)$ is the resultant state of $V(s, i)$ after i has performed action act_i . Let us consider the case when act_i is $Copy_{i,\alpha,\beta} \in T_i(s)$ for some $\alpha \in \Omega$. Since the rule $Copy_{i,\alpha,\beta}$ is applicable at s , $\alpha \notin V(s, i)$ while $\alpha \in V(s, j)$ for $j \in A_g$ and $i \neq j$. Therefore, there exists a MCS (Maximal Consistent Set) χ such that $\chi \supseteq V(s, i)$, and $cp_i^{\bar{m}} \wedge \neg B_i \alpha \wedge B_j \alpha \in \chi$, for some $m \in \{0, \dots, n_C(i)\}$ and $|V^M(s, i)| \leq n_M(i)$. By Axiom A6 and MP (Modus Ponens), $EX(B_i \alpha \wedge cp_i^{\bar{m}+1}) \in \chi$. Therefore, according to the construction, $\exists s' \in S$ such that $s T s'$, $V(s', i) \subseteq \chi'$ for some χ' , and $B_i \alpha \wedge cp_i^{\bar{m}+1} \in \chi'$. Therefore, $V(s', i) = V(s, i) \setminus \{\beta\} \cup \{\alpha\} \cup \{cp_i^{\bar{m}+1}\}$.

¹Definition 4.9

- Axiom A7 states that at most one new formula can be added. we need to prove that for all $s \in S$, $act_i \in T_i(s)$ and $i \in A_g$, there exists a $s' \in S$ such that $s T s'$ and $V(s', i)$ is the resultant state of $V(s, i)$ after i has performed action act_i . By axioms A7, $V(s', i)$ is different from $V(s, i)$ by at most one formula added and possibly a formula is removed. Let us consider the case when act_i is $Rule_{i,r,\gamma} \in T_i(s)$ for some $r \in \mathfrak{R}$. Since the rule $Rule_{i,r,\gamma}$ is applicable at s , therefore $V(s', i) = V(s, i) \setminus \{\gamma\} \cup \{\alpha\} \cup \{\beta\}$. If no formula is added or removed, we consider act_i to be $Idle_i$.
- For the axiom A8, a formula $\alpha \in \Omega$ is considered as an initial fact which is permanently stored in the static memory which means $\alpha \in V(s, i)$, there exists $s' \in S$ such that $s T s'$ and $V(s', i)$ is the resultant state of $V(s, i)$. Similarly, the rules of an agent i always resides in the static memory which can never be deleted.
- Let us now consider the case where a formula α is added by one of the valid reasoning actions. By axiom A9, if $cp_i^{\bar{m}} \in V(s, i)$ for some $m \in \{0, \dots, n_C(i)\}$ then either $cp_i^{\bar{m}}$ or $cp_i^{\bar{m}+1} \in V(s', i)$. If $cp_i^{\bar{m}} \in V(s', i)$ then set act_i to be $Rule_{i,r,\beta}$ for some $r \in V(s, i)$, $\alpha = cons(\bar{r}) \notin V(s, i)$. If $cp_i^{\bar{m}+1} \in V(s', i)$, then set act_i to be $Copy_{i,\alpha,\beta}$. Otherwise act_i to be $Idle_i$.
- For A10, we say at the root s_0 of (S, T) , the construction of the model implies that there exists a maximally consistent set (MCS) χ_0 such that $\chi_0 \supseteq V(s_0, i)$ and $start \in \chi_0$. Therefore, it is trivial that $cp_i^{\bar{0}} \in V(s_0, i)$.
- A11 and A12 are already defined by agent's beliefs. For any rule $r \in \mathfrak{R}$, $i \in A_g$ and $s, s' \in S$, we say $r \in V(s, i)$ iff $r \in V(s', i)$. In our model, agents do not exchange rules as messages.
- A13 is similar to A5 but with idle action transition. In the model, we say for all $s \in S$, $act_i \in T_i(s)$ and $i \in A_g$, there exists a state $s' \in S$ such that $s T s'$ and $V(s', i)$ is the resultant state of $V(s, i)$ after i has performed action act_i . Let us consider the case when act_i is $Idle_i$. Since the rule is applicable, therefore

$$V(s', i) = V(s, i).$$

- For the axiom A14, we can show that for any tuple of actions $\langle act_1, act_2, \dots, act_{n_{A_g}} \rangle$, $act_i \in T_i(s)$ is applicable at $s \in S$, for all $i \in A_g$, then there exists a $s' \in S$ such that $V(s', i)$ is the resultant state of $V(s, i)$ after performing act_i at s by agent i , for all $i \in A_g$.

4.8 \mathcal{L}_{OCRS} Proofs of Correctness

In this section, we describe the satisfiability problem for the model to prove that this model is decidable by formulas of language in $\mathbb{M}(n_m, n_C)$.

Definition 4.5. Satisfiability of \mathcal{L}_{OCRS} formulas

We say that a formula $\varphi \in \Omega$ of the language \mathcal{L} is satisfied in a model $\mathbb{M} = (S, T, V)$ if and only if $\mathbb{M}, s \models \varphi$ for some state $s (= \pi_n) \in S$. For a given class of models \mathbb{C} , a formula φ is said to be valid in \mathbb{C} , written as $\mathbb{C} \models \varphi$ if $\mathbb{M} \models \varphi$ for some $\mathbb{M} \in \mathbb{C}$.

Definition 4.6. Global Satisfiability of \mathcal{L}_{OCRS} formulas

A formula $\varphi \in \Omega$ of \mathcal{L} is said to be globally satisfied in a model $\mathbb{M} = (S, T, V)$, $\mathbb{M} \models \varphi$ if $\mathbb{M}, s \models \varphi$, $\forall s \in S$.

Definition 4.7. Validity of \mathcal{L}_{OCRS} formulas

A formula $\varphi \in \Omega$ is said to be valid in a model \mathbb{M} if and only if it is true for all $s \in S$ in the model.

Definition 4.8. \mathcal{L}_{OCRS} Model Equivalence

For the two given models $\mathbb{M} = (S, T, V)$ and $\mathbb{M}' = (S', T', V')$, we say they are modally equivalent for any $s \in S$ and $s' \in S'$ if and only if $\{ \varphi \mid \mathbb{M}, s \models \varphi \} \equiv \{ \psi \mid \mathbb{M}', s' \models \psi \}$.

For a given logic \mathcal{L}_{OCRS} , we need to prove completeness proofs with respect to some classes to show that every consistent set of formulas can be satisfied in a suitable model.

To build the satisfying model for the completeness proof, we first need to define Maximal consistent set of formulas for the model and to build canonical model which is syntactically constructed from maximally consistent set.

Definition 4.9. Maximal Consistent Set (MCS)

We say that a set of formulas Ω is a maximal consistent set with respect to a language \mathcal{L} of $\mathcal{L}_{O CRS}$ if and only if Ω is \mathcal{L} -consistent, and it becomes \mathcal{L} -inconsistent for any superset of Ω . If Ω is a maximal \mathcal{L} -consistent set of formulas then we say it is a \mathcal{L}_{MCS} .

There are two reasons for the use of maximal consistent set in the completeness proofs.

- First, every state s in the model \mathbb{M} is associated with a set of formulas, formally written as $\{\alpha \mid \mathbb{M}, s \models \alpha\}$. The set of formulas is represented by \mathcal{L}_{MCS} , which means if a formula α is true in the model \mathbb{M} then α belongs to \mathcal{L}_{MCS} .
- Second, if a state s is related to the next state s' in the model \mathbb{M} then we clearly say that the set of formulas included in the MCS associated with s and s' is coherently related.

Based on the second observation, models give rise to accumulation of coherently related MCSs. Now we list some properties of maximal consistent set \mathcal{L}_{MCS} .

Proposition 4.2. *We say if $\mathcal{L}_{O CRS}$ is a logic and Ω is a \mathcal{L}_{MCS} then:*

1. $\varphi \in \Omega$ iff $\Omega \vdash_{\mathbb{M}} \varphi$.
2. $\varphi \vee \psi \in \Omega$ iff $\varphi \in \Omega$ or $\psi \in \Omega$.
3. $\varphi \wedge \psi \in \Omega$ iff $\varphi \in \Omega$ and $\psi \in \Omega$.
4. $\varphi \rightarrow \psi \in \Omega$ iff $\varphi \in \Omega$, then $\psi \in \Omega$.

Lemma 4.1. *Lindenbaum Lemma: For every \mathcal{L} -consistent set Ω there is a maximally consistent superset Ω^+ .*

Proof: Enumerate all formulas $\varphi_1, \varphi_2, \dots$ of the language \mathcal{L} and define an ascending chain of sets of formulas $\Omega^0 \subseteq \Omega' \subseteq \dots \subseteq \Omega^n \subseteq \dots$ such that

$$\Omega^0 = \Omega$$

$$\Omega^{m+1} = \begin{cases} \Omega^m \cup \{\varphi_k\} & \text{if } \varphi_k \text{ is } \mathcal{L}\text{-consistent with } \Omega^m \\ \Omega^m, & \text{otherwise} \end{cases}$$

Let $\Omega^+ = \bigcup_{m \in \mathbb{N}} \Omega^m$, then Ω^+ is both maximal in \mathcal{L} and \mathcal{L} -consistent.

Suppose, Ω^+ is not \mathcal{L} -consistent. Then, $\exists i \geq 1$ (this is because we assume $\Omega^0 (i = 0)$ is \mathcal{L} -consistent) such that Ω^i is \mathcal{L} -inconsistent. Thus Ω^{i-1} must be inconsistent and hence each $\Omega^j (j < i)$ is \mathcal{L} -inconsistent. This contradicts the fact that $\Omega (= \Omega^0)$ is \mathcal{L} -inconsistent. Next let us suppose that Ω^+ is not maximal. Then $\exists \Omega' \supset \Omega^+$ such that Ω' is \mathcal{L} -consistent. Therefore, $\Omega' \supseteq \Omega^+ \cup \{\varphi_k\}$ for some formula φ_k . Since φ_k is \mathcal{L} -consistent with Ω^+ , and $\Omega^{n+1} = \Omega^n \cup \{\varphi_k\}$. Hence Ω' does not exist.

4.8.1 The satisfiability Problem of $\mathbb{L}(n_M, n_C)$

In this section, we describe the satisfiability of formulas of $\mathbb{L}(n_M, n_C)$ in the class $\mathbb{M}(n_M, n_C)$. The satisfiability of formulas is determined by a model \mathbb{M} to see whether it is decidable in $\mathbb{M}(n_M, n_C)$. We also prove the satisfiability of formulas in the canonical model which is satisfied in $\mathbb{M}(n_M, n_C)$.

4.8.1.1 The Canonical Model of $\mathbb{M}(n_M, n_C)$

We say that $\mathbb{M} = (S, T, V)$ is an arbitrary model in $\mathbb{M}(n_M, n_C)$. At a given state $s \in S$, V is a mapping which identifies the state of the internal memory and the record of communication counter for each agent in the system. In addition, the definition of $\mathbb{M}(n_M, n_C)$ determines paths starting from s . For a given different model $\mathbb{M}' = (S', T', V')$ of $\mathbb{M}(n_M, n_C)$ if there is a state s' which is the replica of s , i.e., $V(s, i)$ is the same as $V'(s', i)$ for all $i \in A_g$. Then the set of paths in \mathbb{M}' starting from s' is same as the set of

paths starting from s in \mathbb{M} if we don't differentiate states with the same value of V and V' . This property of models in $\mathbb{M}(n_M, n_C)$ allows us to consider a canonical model in the class $\mathbb{M}(n_M, n_C)$ which includes all models.

Definition 4.10. A canonical model $\mathbb{M}^c = (S^c, T^c, V^c)$ is a triple which is constructed syntactically from maximal consistent sets.

- $S^c = \{\Omega \cup CP\}$ where Ω is the set of all \mathcal{L}_{MCS} and $CP = \bigcup_{i \in A_g} CP_i$
- T^c is a canonical relation; for all states $s^c, s'^c \in S^c$, $s^c T^c s'^c$ if and only if there are $act_i \in T_i^c(s)$ for all $i \in A_g$.
- $V^c(s^c, i) = \{B_i \Gamma \cup cp_i^{\neg n}\}$ where $\forall s^c \in S^c, \Gamma \in \Omega$ and for all $i \in A_g$. V^c is called canonical valuation.

Lemma 4.2. *Canonical Model(CM) is a model of the class $\mathbb{M}(n_M, n_C)$.*

Proof. The conditions given in the model $\mathbb{M}(n_M, n_C)$ follows the definition of canonical model. We explain the following to complete the proof.

$V^c(s^c, i) = \{B_i \Gamma \cup cp_i^{\neg m}\}$; where $\Gamma \in \mathcal{L}$, $\Gamma \subseteq \Omega$ and $|\Gamma| \leq V^c(s^c, i)$ for any $s^c \in S^c$, $i \in A_g$ and for some $m \in \{0, \dots, n_C(i)\}$.

Now we prove the existence lemma in which states are coherently related to define the required accessibility relations. Existence lemma ensures the success of the construction by defining the required coherently related MCSs.

Lemma 4.3. *Existence Lemma: Let $\mathbb{M}^c = (S^c, T^c, V^c)$ be a canonical model for the logic $\mathcal{L}_{O CRS}$. For any formula $\varphi \in \Omega$ and any state $s^c \in S^c$, if $F\varphi \in V^c(s^c, i)$ then there is a state $s'^c \in S^c$ such that $s^c T^c s'^c$ and $\varphi \in V^c(s'^c, i)$, for any $i \in A_g$.*

Proof: We assume $F\varphi \in V^c(s^c, i)$, we need to construct a state s'^c such that $s^c T^c s'^c$ and $\varphi \in V^c(s'^c, i)$. For this, let $\Delta = \{\varphi\} \cup \{\psi \mid F\psi \in V^c(s^c, i)\}$ then Δ is consistent. Suppose Δ is inconsistent, then there are $\psi_1, \dots, \psi_n \in \Delta$ such that $\psi_1 \wedge \dots \wedge \psi_n \vdash_{\mathbb{M}} \neg\varphi$

and it follows that $G(\psi_1 \wedge \dots \wedge \psi_n) \vdash_{\mathbb{M}} G\neg\varphi$. The formula $G(\psi_1 \wedge \dots \wedge \psi_n)$ is represented by deduction theorem as $G\psi_1 \wedge \dots \wedge G\psi_n$. Hence, by propositional calculus, it follows $\vdash_{\mathbb{M}} (G\psi_1 \wedge \dots \wedge G\psi_n) \rightarrow G\neg\varphi$. Now $G\psi_1 \wedge \dots \wedge G\psi_n \in V^c(s^c, i)$ and $V^c(s^c, i)$ is a \mathcal{L}_{MCS} . Then $G\neg\varphi \in V^c(s^c, i)$, it follows that $\neg F\varphi \in V^c(s^c, i)$ because $\neg F\varphi \equiv G\neg\varphi$. But this contradicts our assumption as $F\varphi \in V^c(s^c, i)$. Hence Δ is consistent. By Lindenbaum lemma, Δ can be extended to a \mathcal{L}_{MCS} Δ^+ with $\varphi \in \Delta^+$ and $\{\psi \mid G\psi \in V^c(s^c, i)\} \subseteq \Delta^+$. So, by construction $\varphi \in V^c(s'^c, i)$ and hence $s^c T^c s'^c$.

Now we prove the truth lemma which states that any formula φ belongs to a \mathcal{L}_{MCS} is equivalent to the formula φ which is true in some model \mathbb{M} .

Lemma 4.4. Truth Lemma: Let $\mathbb{M}^c = (S^c, T^c, V^c)$ be a canonical model for the logic $\mathcal{L}_{O CRS}$. For any formula $\varphi \in \Omega$, $\mathbb{M}^c, s^c \models \varphi$ if and only if $\varphi \in V^c(s^c, i)$ where $s^c \in S^c$ and for any $i \in A_g$.

Proof: The proof is given by structural induction on the complexity of φ and the definition of V^c provides the base case. Since $V^c(s^c, i)$ is consistent and maximal while boolean cases are trivial. We assume $\mathbb{M}^c, s^c \models EX \psi$ if and only if there exists a state s'^c such that $s^c T^c s'^c$ and $\mathbb{M}^c, s'^c \models \psi$. So $\psi \in V^c(s'^c, i)$ by the induction hypothesis and $EX\psi \in V^c(s^c, i)$ by the definition of T^c . Now for the reverse direction, the existence lemma guarantees that $EX\psi \in V^c(s^c, i)$ which implies the existence of a state $s'^c \in S^c$ such that $s^c T^c s'^c$ and $\psi \in V^c(s'^c, i)$. By hypothesis $\mathbb{M}^c, s'^c \models \psi$ and hence $\mathbb{M}^c, s^c \models EX\psi$.

Lemma 4.5. Let $\mathbb{M}^c = (S^c, T^c, V^c)$ be a canonical model for the logic $\mathcal{L}_{O CRS}$. For any formula $\varphi \in \Omega$, for all states $s^c, s'^c \in S^c$ and for all $i \in A_g$ if $s^c T^c s'^c$ and $\varphi \in V^c(s'^c, i)$ but $\varphi \notin V^c(s^c, i)$ then

(i) $V^c(s'^c, i) = V^c(s^c, i) \cup \{\varphi\}$ and

(ii) φ must be a literal.

Proof: We assume $s^c T^c s'^c$, $\varphi \in V^c(s'^c, i)$ and $\varphi \notin V^c(s^c, i)$. As the set of initial contexts

for each agent i , $\vdash_{\mathbb{M}} B_i \varphi \rightarrow AX B_i \varphi$ and $V^c(s^c, i) \subseteq V^c(s'^c, i)$. We further assume that axiom A8 entails $\psi \in V^c(s^c, i)$ based on case (i) where for any $\psi \in V^c(s'^c, i)$ and $\psi \neq \varphi$. For case (ii), if φ is not a literal then it must be some rule $r \in \mathfrak{R}$. This is the only case for the given axioms A11 and A12, if $r \in \mathfrak{R}$ then $\varphi \in V(s^c, i)$ contrary to hypothesis.

4.8.2 Bisimulation

Bisimulation is a symmetric simulation for two transition systems. It presents equivalence relations on the structure which satisfy the same formulas and captures state equivalences and process equivalences [Stirling, 2012, Blackburn et al., 2002]. In other words, we say that two different finite structures are bisimilar if and only if they satisfy the set of formulas. In the logic \mathcal{L}_{OIRS} , the model \mathbb{M} and \mathbb{M}^c do not distinguish bisimilar structure. We say that $\mathbb{M} = (S, T, V)$ and $\mathbb{M}^c = (S^c, T^c, V^c)$ are models of the form of ω -tree structure in the \mathcal{L}_{OIRS} . A non-empty binary relation $\mathcal{Z} \subseteq S \times S^c$ is called bisimulation between \mathbb{M} and \mathbb{M}^c , when following conditions are satisfied.

- Invariance: If $\mathcal{Z}(s, s^c)$ then both s and s^c are identical satisfying the same set of formulas. In other words, we say that two states have the same valuation: $V(s, i) = V^c(s^c, i)$
- Forth Condition: If $\mathcal{Z}(s, s^c)$ and $s T s'$ in \mathbb{M} , then there exists a state s'^c in \mathbb{M}^c such that $s^c T^c s'^c$ and $\mathcal{Z}(s', s'^c)$.
- Back Condition: If $\mathcal{Z}(s, s^c)$ and $s^c T^c s'^c$ in \mathbb{M}^c , then there exists a state s' in \mathbb{M} such that $s T s'$ and $\mathcal{Z}(s', s'^c)$

For any formula $\varphi \in \Omega$ of \mathcal{L} , $\mathbb{M}, s \models \varphi$ if and only if $\mathbb{M}^c, s^c \models \varphi$.

Lemma 4.6. *A formula is said to be satisfiable in $\mathbb{M}(n_M, n_C)$ if and only if it is satisfied in the canonical model (CM).*

Proof: Let α be a formula and $\mathbb{M} = (S, T, V)$ is a model in $\mathbb{M}(n_m, n_C)$. Let $V(s, i) = \Gamma_i$, where Γ_i contains the set of formulas at state s of agent i including a communication

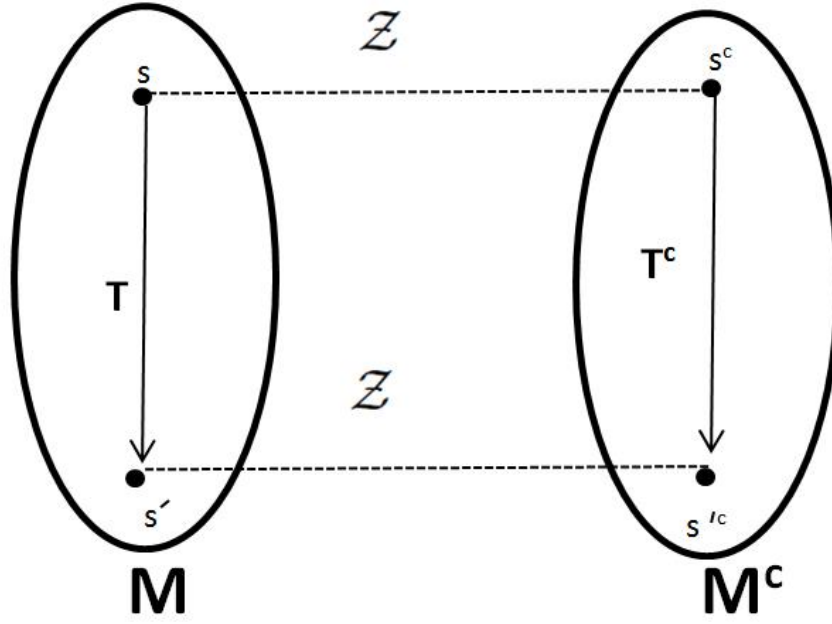


Figure 4.4: Bisimilar Model Forth condition

counter $cp_i^{\bar{m}}$, for all $i \in A_g$ and any $s \in S$. If s corresponds to s^c then we define s^c as $V^c(s^c, i) = \Gamma_i^c \in S^c$. To prove this in canonical model, we inductively define the structure of the formula α for any s of \mathbb{M} , we have $\mathbb{M}, s \models \alpha$ if and only if $\mathbb{M}^c, s^c \models \alpha$.

- If $\alpha = B_i C(a)$ then we have
 - $\mathbb{M}, s \models B_i C(a)$ iff $C(a) \in V(s, i) = \Gamma_i$ which corresponds in CM as $V^c(s^c, i)$

$$\text{iff } C(a) \in V^c(s^c, i)$$

$$\text{iff } \mathbb{M}^c, s^c \models B_i C(a)$$

The set of all ground facts given in Section 4.5.1 are defined in the similar fashion.

- If $\alpha = cp_i^{\bar{m}}$
 - $\mathbb{M}, s \models cp_i^{\bar{m}}$ iff $cp_i^{\bar{m}} \in V(s, i)$ which corresponds in CM as $V^c(s^c, i)$

$$\text{iff } cp_i^{\bar{m}} \in V^c(s^c, i)$$

$$\text{iff } \mathbb{M}^c, s^c \models cp_i^{\bar{m}}$$

- If $\alpha = \neg\varphi$
 - $\mathbb{M}, s \models \neg\varphi$ iff $\mathbb{M}, s \not\models \varphi$
iff $\mathbb{M}^c, s^c \not\models \varphi$ by induction hypothesis ²
iff $\mathbb{M}^c, s^c \models \neg\varphi$
- If $\alpha = \varphi \vee \psi$
 - $\mathbb{M}, s \models \varphi \vee \psi$ iff $\mathbb{M}, s \models \varphi$ or $\mathbb{M}, s \models \psi$
iff $\mathbb{M}^c, s^c \models \varphi$ or $\mathbb{M}^c, s^c \models \psi$ by induction hypothesis
iff $\mathbb{M}^c, s^c \models \varphi \vee \psi$
- If $\alpha = \varphi \wedge \psi$
 - $\mathbb{M}, s \models \varphi \wedge \psi$ iff $\mathbb{M}, s \models \varphi$ and $\mathbb{M}, s \models \psi$
iff $\mathbb{M}^c, s^c \models \varphi$ and $\mathbb{M}^c, s^c \models \psi$ by induction hypothesis
iff $\mathbb{M}^c, s^c \models \varphi \wedge \psi$
- If $\alpha = \varphi U \psi$
 - $\mathbb{M}, s \models \varphi U \psi$ iff $\exists m \geq n$ such that for all $k \in [n, m)$ $\mathbb{M}, s_k \models \varphi$ and $\mathbb{M}, s_m \models \psi$
iff $\exists m \geq n$ such that for all $k \in [n, m)$ $\mathbb{M}^c, s_k^c \models \varphi$ and $\mathbb{M}^c, s_m^c \models \psi$ by induction hypothesis
iff $\mathbb{M}^c, s^c \models \varphi U \psi$
- If $\alpha = EX \varphi$
 - $\mathbb{M}, s \models EX \varphi$ iff there exists $s' \in S$ such that $s T s'$ and $\mathbb{M}, s' \models \varphi$
iff $\exists s'^c \in S^c$ such that $s^c T s'^c$, and $\mathbb{M}^c, s'^c \models \varphi$ by induction hypothesis
iff $\mathbb{M}^c, s'^c \models EX \varphi$

²For induction hypothesis, we say that the base case is $s_0 \in S$ and the induction step states: If $s \in S$ then $s' \in S$.

- If $\alpha = AX \varphi$
 - $\mathbb{M}, s \models AX \varphi$ iff for any path (s_0, s_1, \dots) starting from s (i.e., $s_0 = s$),
 $\exists k \geq 0$ such that $\mathbb{M}, s_k \models \varphi$
 iff $\mathbb{M}^c, s_k^c \models \varphi$ by induction hypothesis
 iff $\mathbb{M}^c, s^c \models AX \varphi$

4.8.3 Soundness Proofs in the Canonical Model \mathbb{M}^c

In this section, we provide the soundness proofs to show the validity of axioms in the canonical model. As given in lemma 4.6, a formula φ is said to be satisfiable in the model $\mathbb{M}(n_M, n_C)$ if and only if it is satisfied in the canonical model $\mathbb{M}^c(n_M, n_C)$. Similarly, we say a formula φ is valid in the model $\mathbb{M}(n_M, n_C)$ if and only if it is valid on $\mathbb{M}^c(n_M, n_C)$. These proofs are given below:

- For the axiom A2 in canonical model \mathbb{M}^c , let $\phi = \bigwedge_{\alpha \in D_M(i)} B_i \alpha \rightarrow \neg B_i \beta$ where $i \in A_g$ and $\alpha \in \Omega$ and $D_M(i) \subseteq \Omega$ such that $|D_M(i)| = n_M(i)$. Let $s^c \in S^c$ be an arbitrary state and there exists $\beta \in \Omega$ such that $\beta \notin V^c(s^c, i)$. Then, we have that $\mathbb{M}^c, s^c \not\models B_i \beta$, which implies that $\mathbb{M}^c, s^c \models \bigwedge_{\alpha \in D_M(i)} B_i \alpha$. Therefore, $\mathbb{M}^c, s^c \models \bigwedge_{\alpha \in D_M(i)} B_i \alpha \rightarrow \neg B_i \beta$. Hence, $\vdash_{\mathbb{M}^c} \bigwedge_{\alpha \in D_M(i)} B_i \alpha \rightarrow \neg B_i \beta$. Thus ϕ is a formula in any state s^c of \mathbb{M}^c and it is valid.
- Let $cp_i^{\bar{m}}$ be a communication counter formula and s^c is an arbitrary state in S^c such that $cp_i^{\bar{m}} \in V^c(s^c, i)$ for some $m \in \{0, \dots, n_C(i)\}$, then $\mathbb{M}^c, s^c \models cp_i^{\bar{m}}$. This means the formula is true at any state s^c of \mathbb{M}^c . Hence, $\vdash_{\mathbb{M}^c} \bigvee_{l=0, \dots, n_C(i)} cp_i^{\bar{m}}$ is valid in \mathbb{M}^c .
- For the axiom A4 in CM, we assume $cp_i^{\bar{n}} \rightarrow \neg cp_i^{\bar{m}}$ be a formula and s^c is an arbitrary state in S^c , $m \neq n$ and for all $i \in A_g$. If $\mathbb{M}^c, s^c \models cp_i^{\bar{n}}$ then $cp_i^{\bar{n}} \in V^c(s^c, i)$. As a result, $cp_i^{\bar{m}} \notin V^c(s^c, i)$ which means $\mathbb{M}^c, s^c \not\models cp_i^{\bar{m}}$ where $s^c \in S^c$.

Therefore, $\mathbb{M}^c, s^c \models cp_i^{\bar{n}} \rightarrow \neg cp_i^{\bar{m}}$. So we say that this axiom is true in any state s^c of \mathbb{M}^c . Hence, $\vdash_{\mathbb{M}^c} cp_i^{\bar{n}} \rightarrow \neg cp_i^{\bar{m}}$ is valid in \mathbb{M}^c .

- For the axiom A5 in CM, we assume $\mathbb{M}^c, s^c \models B_i r \wedge \bigwedge_{P \in \text{ant}(\bar{r})} B_i P \wedge cp_i^{\bar{m}} \wedge \neg B_i \text{cons}(\bar{r})$ where $s^c (= \pi_n^c) \in S^c$ and $|V^c(s^c, i)| \leq n_M(i)$. If the action is $Rule_{i,r,\beta}$ then by axiom A5 and according to the definition of T^c in \mathbb{M}^c , there exists a state $s'^c \in S^c$ such that $s^c T^c s'^c$ and $V^c(s'^c, i) = V^c(s^c, i) \setminus \{\beta\} \cup \{\text{cons}(\bar{r})\}$. Then we have $\mathbb{M}^c, s'^c \models B_i \text{cons}(\bar{r}) \wedge cp_i^{\bar{m}}$. Therefore, we say that this axiom is true in any state s^c of \mathbb{M}^c . Hence, $\vdash_{\mathbb{M}^c} B_i r \wedge \bigwedge_{P \in \text{ant}(\bar{r})} B_i P \wedge cp_i^{\bar{m}} \wedge \neg B_i \text{cons}(\bar{r}) \rightarrow EX(B_i \text{cons}(\bar{r}) \wedge cp_i^{\bar{m}})$ is valid in \mathbb{M}^c .
- Let us say that $s^c (= \pi_n^c)$ is an arbitrary state in S^c , we assume $\mathbb{M}^c, s^c \models cp_i^{\bar{m}} \wedge \neg B_i \alpha \wedge B_j \alpha$ and $|V^c(s^c, i)| \leq n_M(i)$. If the action is $Copy_{i,\alpha,\beta}$ then by axiom A6 and according to the definition of T^c in \mathbb{M}^c , there exists a state $s'^c \in S^c$ such that $s^c T^c s'^c$ and $V^c(s'^c, i) = V^c(s^c, i) \setminus \{\beta, cp_i^{\bar{m}}\} \cup \{\alpha \cup cp_i^{\bar{m}+1}\}$ where $\beta \in \Gamma$. Then we have $\mathbb{M}^c, s'^c \models B_i \alpha \wedge cp_i^{\bar{m}+1}$. Therefore, this axiom is true in any state s^c of \mathbb{M}^c . Hence, $\vdash_{\mathbb{M}^c} cp_i^{\bar{m}} \wedge \neg B_i \alpha \wedge B_j \alpha \rightarrow B_i \alpha \wedge cp_i^{\bar{m}+1}$ which is valid in \mathbb{M}^c .
- For the axiom A7 in the canonical model, let us say $\mathbb{M}^c, s^c \models EX(B_i \alpha \wedge B_i \beta)$ where $\alpha \notin V^c(s^c, i)$, $\beta \notin V^c(s^c, i)$ for any $i \in A_g$, and $|V^c(s^c, i)| \leq n_M(i)$. If the action is $Rule_{i,r,\gamma}$ then by axiom A7 and according to the definition of T^c in \mathbb{M}^c , there exists a state $s'^c \in S^c$ such that $s^c T^c s'^c$ and $V^c(s'^c, i) = V^c(s^c, i) \setminus \{\gamma\} \cup \{\alpha\} \cup \{\beta\}$ where $\alpha, \beta \in \Omega$. Then we have that $\mathbb{M}^c, s'^c \models B_i \alpha$ or $\mathbb{M}^c, s'^c \models B_i \beta$. Therefore, this axiom is true in any state s^c of \mathbb{M}^c . Hence, $\vdash_{\mathbb{M}^c} EX(B_i \alpha \wedge B_i \beta) \rightarrow B_i \alpha \vee B_i \beta$ is valid in \mathbb{M}^c .
- For the axiom A8 in CM, Let us say $\mathbb{M}^c, s^c \models B_i \alpha$ where $\alpha \in V^c(s^c, i)$. By the definition of T^c in \mathbb{M}^c , if there is a state $s'^c \in S^c$ such that $s^c T^c s'^c$ and $V^c(s'^c, i) = V^c(s^c, i)$. Then we have $\mathbb{M}^c, s'^c \models B_i \alpha$. Therefore, this axiom is true in any state s^c of \mathbb{M}^c . Hence, $\vdash_{\mathbb{M}^c} B_i \alpha \rightarrow AX B_i \alpha$ which is valid in \mathbb{M}^c .

- Let us suppose $\mathbb{M}^c, s^c \models EX(B_i\alpha \wedge cp_i^{\bar{m}})$ where for any $\alpha \in \Omega$, $EX(B_i\alpha \wedge cp_i^{\bar{m}}) \in V^c(s^c, i)$, $|V^c(s^c, i)| \leq n_M(i)$ and for all $i \in A_g$. By axiom A9 and according to the definition of T^c in \mathbb{M}^c , there exists a state $s'^c \in S^c$ such that $s^c T^c s'^c$. In this case, the formula α can be added by one of the valid reasoning actions. Then $V^c(s'^c, i) = V^c(s^c, i) \cup \{\alpha\} \cup ByRule_i(\alpha, m) \cup ByCopy_i(\alpha, m)$. Therefore, we say that this axiom is true in any state s^c of \mathbb{M}^c . Hence, $\vdash_{\mathbb{M}^c} EX(B_i\alpha \wedge cp_i^{\bar{m}}) \rightarrow \alpha \vee ByRule_i(\alpha, m) \vee ByCopy_i(\alpha, m)$ is valid in \mathbb{M}^c .
- For the axioms A11 in canonical model, agent i believes the rule $r \in \mathfrak{R}$ at state $s^c \in S^c$ where $B_i r \in V^c(s^c, i)$.
- For the axioms A12 in canonical model, agent i believes only on its own rule at state $s^c \in S^c$ where $r \notin \mathfrak{R}$ and $\neg B_i r \in V^c(s^c, i)$.
- For the axiom A13 in CM, we assume $\mathbb{M}^c, s^c \models \varphi$ where $s^c (= \pi_n^c) \in S^c$, $\varphi \in V^c(s^c, i)$. If the action is *Idle* then by axiom A13 and according to the definition of T^c in \mathbb{M}^c , there exists a state $s'^c \in S^c$ such that $s^c T^c s'^c$ and $V^c(s'^c, i) = V^c(s^c, i)$. Then we have $\mathbb{M}^c, s'^c \models EX\varphi$. Therefore, we say that this axiom is true in any state s^c of \mathbb{M}^c . Hence, $\vdash_{\mathbb{M}^c} \varphi \rightarrow EX\varphi$ is valid in \mathbb{M}^c .
- We show the validity of axiom A14 in \mathbb{M}^c for any tuple of actions $\langle act_1, act_2, \dots, act_{n_{A_g}} \rangle$, let us say $s^c \in S^c$, $act_i \in T_i^c(s)$ is applicable at s^c , for all $i \in A_g$, then $\exists s'^c \in S^c$ such that $V^c(s'^c, i)$ is the resultant state of $V^c(s^c, i)$.

4.8.4 Completeness Proofs in the Canonical Model \mathbb{M}^c

Theorem 4.3. A logic \mathcal{L}_{OCS} is complete with respect to $\mathbb{M}(n_M, n_C)$ if for any formula $\varphi \in \Omega$ and any set of formulas $\Omega \in \mathcal{L}$, if $\Omega \models_{\mathbb{M}} \varphi$ then $\Omega \vdash_{\mathbb{M}} \varphi$.

Proof. We first see if Ω is \mathcal{L} -inconsistent then its outcome becomes trivial. We consider the case when Ω is \mathcal{L} -consistent. Then we expand Ω to a \mathcal{L}_{MCS} Ω^+ using Lindenbaum lemma, as Ω^+ is maximal in \mathcal{L} and \mathcal{L} -consistent. We construct a canonical model $\mathbb{M}^c =$

(S^c, T^c, V^c) and using truth lemma, it follows that $\mathbb{M}^c, \Omega^+ \models_{\mathbb{M}} \Omega$. It shows that canonical model \mathbb{M}^c is satisfied iff the model \mathbb{M} is satisfied.

According to the lemma 4.6, we say a formula φ is satisfiable in the model $\mathbb{M}(n_M, n_C)$ is also satisfiable in the canonical model, which shows the satisfiability of any consistent set of formulas in CM.

- Let us say α is a formula of the form $B_i\alpha$ where $\alpha \in \Omega$ for all $i \in A_g$. If $\alpha \in D_M(i)$ and $\beta \notin D_M(i)$, then $B_i\alpha$ is one of the formulas in $V^c(s^c, i)$. Therefore, $\bigwedge_{\alpha \in D_M(i)} B_i\alpha \in V^c(s^c, i)$. Hence, $\vdash_{\mathbb{M}^c} \bigwedge_{\alpha \in D_M(i)} B_i\alpha \rightarrow \neg B_i\beta$
- We assume $cp_i^{\bar{m}}$ is a primitive formula where $i \in A_g$ and $m \in \{0, \dots, n_C(i)\}$. If $cp_i^{\bar{m}} \in V^c(s^c, i)$ then $cp_i^{\bar{m}}$ is one of the formulas in Γ_i^c , as $\Gamma_i^c \subseteq V^c(s^c, i)$. So according to propositional tautologies, we say that $\vdash_{\mathbb{M}^c} \Gamma_i^c \rightarrow cp_i^{\bar{m}}$. On the contrary, we say if $cp_i^{\bar{m}} \notin V^c(s^c, i)$ then $cp_i^{\bar{n}}$ is one of the formulas in Γ_i^c for some $m \neq n$. Therefore, by axiom A4 and propositional tautologies, we have $\vdash_{\mathbb{M}^c} \Gamma_i^c \rightarrow \neg cp_i^{\bar{m}}$.
- For the axiom A5 in CM, let us consider the case when act_i is $Rule_{i,r,\beta} \in T_i^c(s)$ for some $r \in \mathfrak{R}$. By axiom A5 and the truth lemma, we say that the rule $Rule_{i,r,\beta}$ is applicable at $s^c, B_i r \wedge \bigwedge_{P \in ant(\bar{r})} B_i P \wedge cp_i^{\bar{m}} \subseteq V^c(s^c, i)$ whereas $B_i cons(\bar{r}) \wedge cp_i^{\bar{m}} \notin V^c(s^c, i)$ and $i \in A_g$. So, by the existence lemma which guarantees a successor state s'^c such that $s^c T^c s'^c$ and then $B_i cons(\bar{r}) \wedge cp_i^{\bar{m}} \in V^c(s'^c, i)$. So by lemma 4.5, $V^c(s'^c, i) = V^c(s^c, i) \setminus \beta \cup \{cons(\bar{r})\}$.
- Let us consider $cp_i^{\bar{m}} \wedge \neg B_i\alpha \wedge B_j\alpha \rightarrow B_i\alpha \wedge cp_i^{\bar{m}+1}$ where act_i is $Copy_{i,\alpha,\beta} \in T_i^c(s)$ for some $\alpha \in \Omega$. By axiom A6 and the truth lemma, we say that the rule $Copy_{i,\alpha,\beta}$ is applicable at $s^c, \alpha \notin V(s^c, i)$ while $\alpha \in V(s^c, j), i \neq j$ and $i, j \in A_g$. So, the existence lemma guarantees a successor state s'^c such that $\alpha \wedge cp_i^{\bar{m}+1} \in V^c(s'^c, i)$. Therefore by lemma 4.5, $V^c(s'^c, i) = V^c(s^c, i) \setminus \{\beta\} \cup \{\alpha\} \cup \{cp_i^{\bar{m}+1}\}$.

- Now consider the case of axiom A7 and by the truth lemma, we say that $EX(B_i\alpha \wedge B_i\beta) \in V^c(s^c, i)$. Since the rule $Rule_{i,r,\gamma}$ is applicable at s^c and the existence lemma guarantees a successor state s'^c such that $s^c T^c s'^c$. Then $V^c(s'^c, i) = V^c(s^c, i) \setminus \gamma \cup \{\alpha\} \cup \{\beta\}$.
- Let us suppose $s^c T^c s'^c$ for all $s^c, s'^c \in S^c$ in \mathbb{M}^c . Then, by the definition of T^c , we say if $\{B_i\alpha | AX B_i\alpha \in V^c(s^c, i)\} \subseteq V^c(s'^c, i)$ then $V^c(s'^c, i) = V^c(s^c, i)$ for any $i \in A_g$.
- We assume $s^c T^c s'^c$ for all $s^c, s'^c \in S^c$ in \mathbb{M}^c . By the axiom A9 and the truth lemma, we say either one of the valid reasoning action is applicable at $V^c(s^c, i)$. Let us consider the case when the action $Rule_{i,r,\beta}$ is applicable at s^c , $ant(\bar{r}) \subseteq V^c(s^c, i)$ whereas $cons(\bar{r}) \notin V^c(s^c, i)$ and $i \in A_g$. And if the action $Copy_{i,\alpha,\beta}$ is applicable at s^c , $\alpha \notin V^c(s^c, i)$ while $\alpha \in V^c(s^c, j)$. Then, by the existence lemma which guarantees the successor state s'^c . Therefore, either $\alpha \wedge cp_i^{\bar{m}} \in V^c(s'^c, i)$ or $\alpha \wedge cp_i^{\bar{m}+1} \in V^c(s'^c, i)$. Therefore by lemma 4.5, $V^c(s'^c, i) = V^c(s^c, i) \setminus \beta \cup \{\alpha\}$. For the action $Idle$, $V^c(s'^c, i) = V^c(s^c, i)$.
- For axiom A13, we assume $s^c T^c s'^c$ for all $s^c, s'^c \in S^c$ in \mathbb{M}^c . Then, by the definition of T^c , we say if $\{\varphi | EX\varphi \in V^c(s^c, i)\} \subseteq V^c(s'^c, i)$ then $V^c(s'^c, i) = V^c(s^c, i)$ for any $i \in A_g$.
- For axiom A14, we say that for any tuple of actions $\langle act_1, act_2, \dots, act_{nA_g} \rangle$, if the action $act_i \in T_i^c(s)$ is applicable at $V^c(s^c, i) \in S^c$ and $\exists s'^c \in S^c$ such that $s^c T^c s'^c$ then $V^c(s'^c, i)$ is the resultant state of $V^c(s^c, i)$ after performing act_i at s^c by agent i , and for all $i \in A_g$.

4.9 Encoding and Verification of $\mathcal{L}_{O CRS}$ Model

In [Rakib et al., 2014], we have shown how to encode and verify some system properties using Maude LTL model checker. We present and discuss Maude encoding and verifica-

tion in Chapter 6. We chose the Maude LTL model checker because it can model check systems whose states involve arbitrary algebraic data types. The only assumption is that the set of states reachable from a given initial state is finite. Rule variables can be represented directly in the Maude encoding, without having to generate all ground instances resulting from possible variable substitutions.

4.10 Conclusion

In this chapter, we have presented a formal logical framework, which is an extension of the frameworks developed by [Alechina et al., 2006, Alechina et al., 2009a, Alechina et al., 2009c] for modelling and verifying context-aware multi-agent systems. Where agents reason using ontology-driven first order Horn-clause rules. We considered space requirement for reasoning in addition to the time and communication resources. We extend CTL^* with belief and communication modalities, and the resulting logic \mathcal{L}_{OCRS} allows us to describe a set of rule-based reasoning agents with bound on time, memory and communication. The key idea underlying the logical approach \mathcal{L}_{OCRS} of context-aware systems is to define a formal logic that axiomatizes the set of transition systems, and it is then used to state various qualitative and quantitative properties of the systems. For example, a qualitative property could be “*Can an agent derives the context φ eventually*”, and quantitative properties could be “*an agent will always derive the context φ in n time steps while exchanging fewer than m messages*” or “*every request of an agent i will be responded by agent j in n time steps*”, among others.

In the next chapter, we would like to present a framework that will allow us to design context-aware agents considering non-monotonic reasoning. The logic developed in this chapter is based on monotonic reasoning where beliefs of an agent cannot be revised based on some contradictory evidence. We extend this logic considering defeasible reasoning which is a simple rule-based technique used to reason with incomplete and inconsistent information.

Chapter 5

The Logic \mathcal{L}_{DROCS}

5.1 Chapter Objectives

- To develop a non-monotonic reasoning based logical framework \mathcal{L}_{DROCS} for resource-bounded context-aware rule-based agents.
- To prove the correctness of axiomatizations such as soundness and completeness.
- To express resource-bounded properties including non-conflicting contextual properties to be verified for the desired system.
- To illustrate the use of \mathcal{L}_{DROCS} framework using a simple health care case study.

5.2 Motivation for the Logic \mathcal{L}_{DROCS}

In this chapter, we introduce the logic \mathcal{L}_{DROCS} which is an extended version of the logic \mathcal{L}_{OCS} presented in the previous chapter. The logic \mathcal{L}_{OCS} is based on monotonic inference where agent's belief cannot be revised based on some contrary evidence. The logic \mathcal{L}_{DROCS} is based on non-monotonic reasoning formalism, which handles inconsistent context information using non-monotonic reasoning. In the logic \mathcal{L}_{DROCS} , we choose defeasible reasoning which is simple and efficient rule-based reasoning technique to re-

solve conflicting context in the memory of non-monotonic reasoning based context-aware agents. Using defeasible reasoning, we develop the rules selection strategy to prioritize rules in order to suitably model the system. In this formalism, the set of actions performed by non-monotonic reasoning agents is based on rules where non-monotonicity is carefully monitored for each action. In a similar fashion, axiomatization of the logic \mathcal{L}_{DROCS} which include the set of actions triggered by the set of rules might have different interpretation as compared to the logic discussed in the previous chapter due to defeasible rules. To model the ontology-driven non-monotonic context-aware multi-agent system, we construct a simple health care domain in the ontology. We use OWL 2 RL ontologies and Semantic Web Rule Language (SWRL) for context-modelling and rules that enables the construction of a formal system. Using the case study, we verified the number of interesting resource-bounded as well as conflicting contextual properties of the system. We extend the temporal logic CTL^* with belief and communication modalities, and the resulting logic \mathcal{L}_{DROCS} allows us to describe a set of rule-based non-monotonic context-aware agents with bounds on computational (time and space) and communication resources. We provide an axiomatization of the logic and prove it is sound and complete.

5.3 Preliminaries

5.3.1 Non-monotonic Rule-based System

The work presented in Chapter 4 is about resource-bounded context-aware rule-based reasoning, where agents are monotonic reasoners. However, non-monotonic reasoning is more practical and plays vital role in many areas of Artificial Intelligence. In practice, everyday tasks are based on non-monotonic reasoning. There are many non-monotonic reasoning techniques exist in rule-based systems [Antoniou, 2002]. We discuss defeasible reasoning technique in this chapter. We have already discussed monotonic as well as non-monotonic reasoning systems in Chapter 3. Now we briefly present a simple example of an online shopping scenario illustrating the comparison between monotonic and non-

monotonic reasoning [Antoniou, 2002].

Monotonic Reasoning : Once we prove something is true, it is true forever. For example, an online shopping vendor gives a special discount to those customers who have a birthday today. The rules are:

- R1: If a customer has a birthday, then a special discount is given.
- R2: If the customer does not have a birthday, then a special discount is not given.

But what happens if a customer has a birthday today but he/she refuses to disclose it.

Non-monotonic Reasoning : Once we prove something is true, it can be negated based on contrary evidence. The important point is to add new information based on contrary evidence.

- R1: If a customer has a birthday, then a special discount is given.
- R2': If the birthday is not known of a customer, then a special discount is not given.

This solves the problem, but the premise of rule R2' is not within the expressive power of predicate logic. Due to that reason, there is a need for a new kind of rule system to resolve inconsistencies which is discussed in the following section.

5.3.2 Defeasible Reasoning

As we have discussed earlier in Chapter 3, defeasible reasoning has been considered as one of the most successful sub-areas in non-monotonic reasoning to deal with inconsistent and incomplete information due to low computational complexity and its focus on implementability [Antoniou, 2002]. It is a simple rule-based reasoning technique that has been used to reason with incomplete and inconsistent information [Antoniou et al., 2001]. A defeasible theory consists of basic components known as knowledge base [Lam, 2012].

A defeasible logic theory consists of a collection of rules that reason over a set of facts to reach a set of defeasible conclusions. It also supports priorities among rules to resolve conflicts. More formally, a defeasible theory \mathcal{D} is a triple $(\mathcal{R}, \mathcal{F}, \succ)$ where \mathcal{R} is a finite set of rules, \mathcal{F} is a finite set of facts, and \succ is a superiority relation on \mathcal{R} . The superiority relation \succ is often defined on rules with complementary heads and its transitive closure is irreflexive, i.e., the relation \succ is acyclic. Rules are defined over literals, where a literal is either a first-order atomic formula P or its negation $\neg P$. For example, given a literal l , the complement $\sim l$ of l is defined to be P if l is of the form $\neg P$, and $\neg P$ if l is of the form P . In the rules, we assume variables are preceded by a question mark (?) and constants are preceded by a single quote ('). In \mathcal{D} there are three kinds of rules those are often represented using different arrows.

Strict rules: If premises of the rule are true then so is the conclusion. An example of a strict rule can be “A person who has a patient identification number is a patient” which can be written as **r1**: $Person(?p), PatientID(?pid), hasPatientID(?p, ?pid) \rightarrow Patient(?p)$.

Defeasible rules can be defeated by contrary evidence. An example rule can be **r2**: $Patient(?p), hasFever(?p, 'High) \Rightarrow hasSituation(?p, 'Emergency)$. This rule states that if the patient has a high fever then there are provable reasons to declare an emergency situation for him, unless there is other evidence that provides reasons to believe the contrary. For example, a defeasible rule **r3**: $Patient(?p), hasFever(?p, 'High), hasConsciousness(?p, 'Yes) \Rightarrow \sim hasSituation(?p, 'Emergency)$. We can observe that the defeasible rule **r3** is more specific (we assume that **r3** is superior to **r2** i.e., $\mathbf{r3} \succ \mathbf{r2}$) and it could override the rule **r2**. That is a defeasible rule is used to represent tentative information that may be used if nothing could be placed against it.

Defeater rules don't support inferences directly, however, they can be used to block the derivation of inconsistent conclusions. Their only use is to prevent conclusions. For example, **r4**: $Patient(?p), hasFever(?p, 'High), hasDBCcategory(?p, 'EstablishedDiabetes) \rightsquigarrow hasSituation(?p, 'Emergency)$.

As usual a rule can have multiple (ground) instances. For example, $Person('Mary), Pa-$

$tientID('P001)$, $hasPatientID('Mary, 'P001) \rightarrow Patient('Mary)$ could be one possible instance of the rule **r1**. In the above rules, suppose the superiority relation \succ among the rules are defined as follows **r1** \succ **r4**, **r4** \succ **r3**, **r3** \succ **r2** and the current set of facts (contexts) are $Person('Mary)$, $PatientID('P001)$, $hasPatientID('Mary, 'P001)$, $hasFever('Mary, 'High)$, $hasConsciousness('Mary, 'Yes)$ then by matching and firing those rules a defeasible conclusion $\sim hasSituation('Mary, 'Emergency)$ can be inferred.

5.3.3 Semantic Context Retrieval for \mathcal{L}_{DROCS}

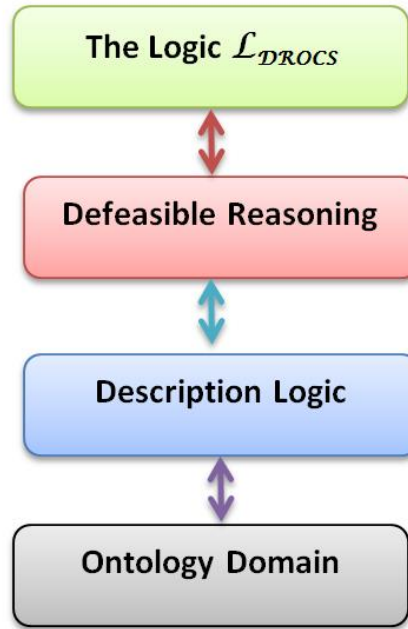


Figure 5.1: Semantic Context Retrieval for \mathcal{L}_{DROCS}

Defeasible logic is an expressive formal language suitable for integrating defeasible reasoning with description logic based ontology because both share a focus on efficiency. The main reason behind the integration is its enhanced reasoning capability and expressability of the ontology based system. The detailed discussion has given in Chapter 3. Figure 5.1 shows the logic flow for the \mathcal{L}_{DROCS} which has synergistic effect in order to solve complex real life problem domains. The logic \mathcal{L}_{DROCS} uses defeasible reasoning rule system which are modelled using description logic based ontology. The knowledge

base construction mechanism is discussed in Chapter 6, which describes the step-by-step process of how a knowledge-base is effectively constructed from an ontological domain in terms of a set of Horn-clause rules.

5.3.4 Semantic Context Modelling

Description logic and ontology have been discussed in Chapter 3. We also provided a comprehensive note on context modelling in the previous chapter. For modelling the system, we have considered a smart space health care monitoring system where OWL concepts and roles capture the relevant information from the domain and represent the desired scenario using a set of logical statements [Esposito et al., 2008]. We model context-aware systems using OWL 2 RL ontologies (and SWRL) and extract rules from an ontology following a similar approach proposed by [Gómez et al., 2007] to design our rule-based non-monotonic context-aware agents. We developed a translator that takes as input an OWL 2 RL ontology in the OWL/XML format (an output file of the Protégé [Protégé, 2011] editor) and translates it to a set of plain text rules. We use the OWL API [Horridge and Bechhofer, 2009] to parse the ontology and extract the set of axioms and facts. The design of the OWL API is directly based on the OWL 2 Structural Specification and it treats an ontology as a set of axioms and facts which are read using the visitor design pattern. We also extract the set of SWRL rules using the OWL API which are already in the Horn-clause rule format. First, atoms with corresponding arguments associated with the head and the body of a rule are identified and we then generate a plain text Horn-clause rule for each SWRL rule using these atoms. Abox axioms are already in Horn-clause formats as well and they are simply rules with empty bodies.

5.4 Context-aware Systems as Multi-agent Defeasible Reasoning Systems

Based on the literature discussed in the previous sections, we incorporate defeasible reasoning on top of the ontology. We model a context-aware system as a multi-agent defeasible reasoning system which consists of $n_{Ag} (\geq 1)$ individual agents $A_g = \{1, 2, \dots, n_{Ag}\}$. Each agent $i \in A_g$ is represented by a triple $(\mathfrak{R}, \mathcal{F}, \succ)$, where \mathcal{F} is a finite set of facts contained in the working memory, $\mathfrak{R} = (\mathfrak{R}^s, \mathfrak{R}^d)$ is a finite set of strict and defeasible rules representing the knowledge base, and \succ is a superiority relation on \mathfrak{R} . Rules are of the form $P_1, P_2, \dots, P_n \hookrightarrow P$ (derived from OWL 2 RL and SWRL with possible user annotation), and a working memory contains ground atomic facts (contexts) taken from ABox representing the initial state of the system.

Without loss of generality, in the rest of this logic we assume \hookrightarrow as either \rightarrow or \Rightarrow . In a rule instance, the antecedents P_1, P_2, \dots, P_n and the consequent P are context information. The antecedents of a rule instance form a complex context which is a conjunction of n contexts. We say that two contexts are contradictory iff they are complementary with respect to \sim , for example, $hasSituation('Mary, 'Emergency)$ and $\sim hasSituation('Mary, 'Emergency)$ are contradictory contexts. The important point to note that in our model, the set of facts transformed from the ABox needs to be consistent, i.e., if it contains pair of contradictory contexts then they can be detected and removed. We assume that the set \mathfrak{R}^s of strict rules is non-contradictory which is used to represent non-defeasible contextual information, however, the set \mathfrak{R}^d of defeasible rules is contradictory and hence the set \mathfrak{R} which is $\mathfrak{R}^s \cup \mathfrak{R}^d$ may also be contradictory.

Conflicting contexts may be resolved using the superiority relation \succ among rules. An agent i can fire the instance of strict rules to infer new non-contradictory contexts, while a defeasible context P can be inferred if there is a rule instance whose consequence is P and there does not exist a stronger rule instance whose consequence is $\sim P$. Since the

translated rules from the ontology are not prioritized. We assume that the rule priorities are fixed by the system designers depending on the intended applications. We further assume that the rule priorities are static. So, the rule firing constraint does not change during the reasoning process. In addition, there are different categories of conflicting rules based on the system specification. Each agent in the system has a reasoning strategy (or conflict resolution strategy) to determine the order in which rules are applied. In case if the rules priority is same and no conflicting rule matches then the random rule is selected to be fired. The concept of communication among agents is similar to \mathcal{L}_{OCS} discussed in Chapter 4

5.5 The Logic \mathcal{L}_{DROCS}

We now introduce the logic \mathcal{L}_{DROCS} in this section. Our approach is based on the work of [Gómez et al., 2007] who show that a subset of *DL* languages can be effectively mapped into a set of strict and defeasible rules. Intuitively the set of translated rules corresponds to the *ABox* joined with *TBox* axioms of an OWL 2 RL ontology. In addition, as mentioned in Chapter 3, we express more complex rule-based concepts using SWRL which allow us to write rules using OWL concepts.

We define the internal language \mathcal{L} of each agent in the system. Let the set of agents be $A_g = \{1, 2, \dots, n_{A_g}\}$, $\mathcal{C} = \{C_1, C_2, \dots, C_l\}$ be a finite set of concepts, $\mathcal{R} = \{R_1, R_2, \dots, R_m\}$ be a finite set of roles. We also define a set $\mathcal{Q} = \{Ask(i, j, P), Tell(i, j, P)\}$, where $i, j \in A_g$ and $P \in \mathcal{C} \cup \mathcal{R}$. Let \mathfrak{R}^s be a finite set of strict rules and \mathfrak{R}^d be a finite set of defeasible rules. Let $\mathfrak{R} = \mathfrak{R}^s \cup \mathfrak{R}^d = \{r_1, r_2, \dots, r_n\}$ be a finite set of rules of the form $P_1, P_2, \dots, P_t \hookrightarrow P$, where $t \geq 0$, $P_i, P \in \mathcal{C} \cup \mathcal{R} \cup \mathcal{Q}$ for all $i \in \{1, 2, \dots, t\}$, $P_i \neq P_j$ for all $i \neq j$, and \hookrightarrow as either \rightarrow or \Rightarrow . More specifically, P_i and P are OWL atoms of the following form: $C_i(x)$ and $R_j(y, z)$. Where $C_i \in \mathcal{C}$, and x is either a variable, an individual or a data value. $R_j \in \mathcal{R}$, when it is an object property y, z are either variables, individuals or data values, however, y is variable or individual and z is a data value when

R_j is a datatype property .

```

Rule ::= Atoms '↪' Atom | ~ Atom
Atoms ::= Atom {, Atom}*
Atom ::= standardAtom | communicationAtom
standardAtom ::= description('i-object ')
                | individualvaluedProperty('i-object ',' i-object ')
                | datavaluedProperty('i-object ',' d-object ')
                | sameIndividuals('i-object ',' i-object ')
                | differentIndividuals('i-object ',' i-object ')
                | dataRange(' d-object ')
                | builtin(' builtinID ',' {d-object}* ')
communicationAtom ::= 'Ask(' i ',' j ',' standardAtom ')
                    | 'Tell(' i ',' j ',' standardAtom ')
i ::= 1 | 2 | ... | nAg
j ::= 1 | 2 | ... | nAg
builtinID ::= URIreference
i-object ::= i-variable | individualID
d-object ::= d-variable | dataLiteral
i-variable ::= 'I-variable('URIreference')'
d-variable ::= 'D-variable('URIreference')'

```

Listing 5.1: Abstract syntax of rules

The Listing 5.1 specifies the abstract syntax of rules using a BNF (Backus-Naur Form). In this notation, the terminals are quoted, the non-terminals are not quoted, alternatives are separated by vertical bars, and components that can occur zero or more times are enclosed braces followed by a superscript asterisk symbol ($\{\dots\}^*$). A class atom represented by $\text{description}(\text{i-object})$ in the BNF consists of an OWL 2-named class and a single argument representing an OWL 2 individual, for example an atom $\text{Patient}(\text{p})$ holds if p is an instance of the class $\text{description Patient}$. Likewise an individual object property atom represented by $\text{individualvaluedProperty}(\text{i-object}, \text{i-object})$ consists of an OWL 2 object property and two arguments representing OWL 2 individuals, for example substituting an individual object property atom, we say that $\text{hasFever}(\text{'Mary'}, \text{'High'})$ holds if Mary has fever as High by a property hasFever . In the same fashion, a data property atom represented by $\text{datavaluedProperty}(\text{i-object}, \text{d-object})$ consists two arguments representing OWL 2 individuals as object value and data value. For example by sbtutiting a data property atom, $\text{hasSituation}(\text{'Mary'}, \text{'Emergency'})$ holds if Mary has situation as Emergency by a property hasSituation and so on.

For convenience, we use the notation $\text{ant}(r)$ for the set of antecedents of r and $\text{cons}(r)$ for the consequent of r , where $r \in \mathfrak{R}$. We fix a finite set of variables X and a finite set

of constants D and assume δ is some substitution function from the set of variables of a rule into D . We denote by $\mathcal{G}(\mathfrak{R})$ the set of all the ground instances of the rules occurring in \mathfrak{R} , which is obtained using δ (more formal definition is given in Definition 5.2). Thus $\mathcal{G}(\mathfrak{R})$ is finite. Let $\bar{r} \in \mathcal{G}(\mathfrak{R})$ be one of the possible instances of a rule $r \in \mathfrak{R}$. $C(a)$, $R(a, b)$, $Ask(i, j, C(a))$, $Ask(i, j, R(a, b))$, $Tell(i, j, C(a))$, and $Tell(i, j, R(a, b))$ are ground atoms, for all $C \in \mathcal{C}$, $R \in \mathcal{R}$. The internal language \mathcal{L} includes all the ground atoms and rules. Let us denote the set of all formulas (rules and ground atoms) by Ω which is finite. In the language \mathcal{L} we have belief operator B_i for all $i \in A_g$. The meaning of belief operator reflects the purpose for which it is designed, for example; we say that $B_i\alpha$ is true if the formula α is in agent i 's memory.

5.5.1 Communication Bound

We assume that there is a bound on communication for each agent i which limits agent i to at most $n_C(i) \in \mathbb{Z}^*$ messages. Each agent has a communication counter, $cp_i^{\bar{n}}$, which starts at 0 ($cp_i^{\bar{0}}$) and is not allowed to exceed the value $n_C(i)$.

For the communication bound, we define the following set:

$$CP_i = \{cp_i^{\bar{n}} | n = \{0, \dots, n_C(i)\}\},$$

$$CP = \bigcup_{i \in A_g} CP_i.$$

5.5.2 Memory Bound and Inconsistent Memory Manipulation

To solve a particular problem, the space (memory cells) available for any given proof is bounded by the size of agent's memory. We divide agent's memory into two parts as rule memory (knowledge base) and working memory. Rule memory holds set of rules, whereas the facts are stored in the agent's working memory. Working memory of an agent i is divided into static memory ($S_M(i)$) and dynamic memory ($D_M(i)$). The $D_M(i)$ of each agent $i \in A_g$ is bounded in size by $n_M(i) \in \mathbb{Z}^*$, where one unit of memory corresponds to the ability to store an arbitrary context. The static part contains initial

information to start up the systems, e.g., initial working memory facts, thus its size is determined by the number of initial facts. The dynamic part contains newly derived facts as the system moves. The size of dynamic memory is determined by the maximal number of formulas that must be simultaneously held in the memory. Only facts stored in $D_M(i)$ may get overwritten, and this happens if an agent's memory is full or a contradictory context arrives in the memory (even if the memory is not full). Whenever newly derived context arrives in the memory, it is compared with the existing contexts to see if any conflict arises. If so then the corresponding contradictory context will be replaced with the newly derived context, otherwise an arbitrary context will be removed if the memory is full. Note that unless otherwise stated, in the rest of this chapter we assume that memory means $D_M(i)$.

5.5.3 Syntax

The syntax of \mathcal{L}_{DROCS} includes the temporal operators of CTL^* and is defined inductively as follows:

- \top (tautology) and *start* (a propositional variable which is only true at the initial moment of time) are well-formed formulas (wffs) of \mathcal{L}_{DROCS} ;
- cp_i^n (which states that the value of agent i 's communication counter is n) is a wff of \mathcal{L}_{DROCS} for all $n \in \{0, \dots, n_C(i)\}$ and $i \in A_g$;
- $B_i C(a)$ (agent i believes $C(a)$), $B_i R(a, b)$ (agent i believes $R(a, b)$), and $B_i r$ (agent i believes r) are wffs of \mathcal{L}_{DROCS} for any $C \in \mathcal{C}$, $R \in \mathcal{R}$, $r \in \mathfrak{R}$ and $i \in A_g$;
- $B_k Ask(i, j, C(a))$, $B_k Ask(i, j, R(a, b))$, $B_k Tell(i, j, C(a))$, and $B_k Tell(i, j, R(a, b))$ are wffs of \mathcal{L}_{DROCS} for any $C \in \mathcal{C}$, $R \in \mathcal{R}$, $i, j \in A_g$, $k \in \{i, j\}$, and $i \neq j$;
- If φ and ψ are wffs of \mathcal{L}_{DROCS} , then so are $\neg\varphi$ and $\varphi \wedge \psi$;
- If φ and ψ are wffs of \mathcal{L}_{DROCS} , then so are $X\varphi$ (in the next state φ), $\varphi U\psi$ (φ holds until ψ), $A\varphi$ (on all paths φ).

Other classical abbreviations for \perp , \vee , \rightarrow and \leftrightarrow , and temporal operations: $F\varphi \equiv \top U \varphi$ (at some point in the future φ) and $G\varphi \equiv \neg F \neg \varphi$ (at all points in the future φ), and $E\varphi \equiv \neg A \neg \varphi$ (on some path φ) are defined as usual.

We define priority relation between rules as follows.

Definition 5.1 (Rule priority). Let $pri : \mathfrak{R} \rightarrow N_{\geq 0}$ be a function that assigns each rule a non-negative integer. We define a partial order \succ on \mathfrak{R} such that for any two rules $r, r' \in \mathfrak{R}$ we say that $r \succ r'$ (rule r has priority over r') iff $pri(r) \geq pri(r')$, where \geq is the standard greater-than-or-equal relation on the set of non-negative integers $N_{\geq 0}$.

5.5.4 Semantics

The semantics of \mathcal{L}_{DROCS} is defined by \mathcal{L}_{DROCS} transition systems which essentially corresponds to the ω -tree structure. The state of each agent corresponds to contents of the working memory and the record of communication counter. Let (S, T) be a pair where S is a set and T is a binary relation on S that is total, i.e., $\forall s \in S \cdot \exists s' \in S \cdot sTs'$. (S, T) is a ω -tree frame iff the following conditions are satisfied.

1. S is a non-empty set and T is total;
2. Let $<$ be the strict transitive closure of T , namely $\{(s, s') \in S \times S \mid \exists n \geq 0, s_0 = s_1, \dots, s_n = s' \in S \text{ such that } s_i T s_{i+1} \forall i = 0, \dots, n-1\}$;
3. For all $s \in S$, the past $\{s' \in S \mid s' < s\}$ is linearly ordered by $<$;
4. There is a smallest element called the root, which is denoted by s_0 ;
5. Each maximal linearly $<$ - ordered subset of S is order-isomorphic to the natural numbers.

A branch of (S, T) is an ω -sequence (s_0, s_1, \dots) such that s_0 is the root and $s_i T s_{i+1}$ for all $i \geq 0$. We denote $B(S, T)$ to be the set of all branches of (S, T) . For a branch

$\pi \in B(S, T)$, π_i denotes the element s_i of π and $\pi_{\leq i}$ is the prefix (s_0, s_1, \dots, s_i) of π . A \mathcal{L}_{DROCS} transition system \mathbb{M} is defined as $\mathbb{M} = (S, T, V)$ where

- (S, T) is a ω -tree frame
- $V : S \times A_g \rightarrow \wp(\Omega \cup CP)$; we define the belief part of the assignment $V^B(s, i) = V(s, i) \setminus CP$ and the communication counter part $V^C(s, i) = V(s, i) \cap CP$. We further define $V^M(s, i) = \{\alpha \mid \alpha \in V^B(s, i) \cap D_M(i)\}$ which represents the set of facts stored in the dynamic memory of agent i at state s . V satisfies the following conditions:

1. $|V^C(s, i)| = 1$ for all $s \in S$ and $i \in A_g$.
2. If sTs' and $cp_i^{\bar{n}} \in V(s, i)$ and $cp_i^{\bar{m}} \in V(s', i)$ then $n \leq m$.

- we say that a rule $r : P_1, P_2, \dots, P_n \hookrightarrow P$ is applicable in a state s of an agent i if $ant(\bar{r}) \in V(s, i)$ and $cons(\bar{r}) \notin V(s, i)$. The following conditions on the assignments $V(s, i)$, for all $i \in A_g$, and transition relation T hold in all models:

1. for all $i \in A_g$, $s, s' \in S$, and $r \in \mathfrak{R}$, $r \in V(s, i)$ iff $r \in V(s', i)$. This describes that agent's program does not change.
2. for all $s, s' \in S$, sTs' holds iff for all $i \in A_g$, $V(s', i) = V(s, i) \setminus \{\beta\} \cup \{cons(\bar{r})\} \cup \{Ask(j, i, C(a))\} \cup \{Tell(j, i, C(a))\} \cup \{Ask(j, i, R(a, b))\} \cup \{Tell(j, i, R(a, b))\}$. This describes that each agent i fires a single applicable rule instance of a rule r , or updates its state by interacting with other agents, otherwise its state does not change. Where β may be an arbitrary context or a contradictory context which can be replaced depending on the status of the memory and the newly derived or communicated context.

The truth of a \mathcal{L}_{DROCS} formula at a point n of a path $\pi \in B(S, T)$ is defined inductively as follows:

- $\mathbb{M}, \pi, n \models \top$,

- $\mathbb{M}, \pi, n \models \text{start}$ iff $n = 0$,
- $\mathbb{M}, \pi, n \models B_i \alpha$ iff $\alpha \in V(\pi_n, i)$,
- $\mathbb{M}, \pi, n \models cp_i^{\bar{m}}$ iff $cp_i^{\bar{m}} \in V(\pi_n, i)$,
- $\mathbb{M}, \pi, n \models \neg \varphi$ iff $\mathbb{M}, \pi, n \not\models \varphi$,
- $\mathbb{M}, \pi, n \models \varphi \wedge \psi$ iff $\mathbb{M}, \pi, n \models \varphi$ and $\mathbb{M}, \pi, n \models \psi$,
- $\mathbb{M}, \pi, n \models X\varphi$ iff $\mathbb{M}, \pi, n + 1 \models \varphi$,
- $\mathbb{M}, \pi, n \models \varphi U \psi$ iff $\exists m \geq n$ such that $\forall k \in [n, m)$ $\mathbb{M}, \pi, k \models \varphi$ and $\mathbb{M}, \pi, m \models \psi$,
- $\mathbb{M}, \pi, n \models A\varphi$ iff $\forall \pi' \in B(S, T)$ such that $\pi'_{\leq n} = \pi_{\leq n}$, $\mathbb{M}, \pi', n \models \varphi$.

We now describe conditions on the models. The transition relation T corresponds to the agent's executing actions $\langle act_1, act_2, \dots, act_{n_{Ag}} \rangle$ where act_i is a possible action of an agent i in a given state s . The set of actions that each agent i can perform are:

- $Rule_{i,r,\beta}$: Agent i firing a selected matching rule instance \bar{r} of r and adding $cons(\bar{r})$ to its working memory and removing β ,
- $Copy_{i,\alpha,\beta}$: Agent i copying α from other agent's memory and removing β , where α is of the form $Ask(j, i, P)$ or $Tell(j, i, P)$,
- $Idle_i$: agent i does nothing but moves to the next state.

Intuitively, β may be an arbitrary context which gets overwritten if it is in the agent's dynamic memory $D_M(i)$ or it is a specific context that contradicts with the newly derived context. If agent's memory is full $|V^M(s, i)| = n_M(i)$ then we require that β has to be in $V^M(s, i)$. When the counter value reaches to $n_C(i)$, i cannot perform copy action any more. Furthermore, not all actions are possible in a given state. For example, there may not be any matching rule instance. Note also that only selected matching rule instances can be fired. That is one rule instance may be selected from the conflict set that has

the highest priority. If there are multiple rule instances with the same priority, the rule instance to be executed is selected non-deterministically. More formally, we define rule selection strategy as follows:

Definition 5.2 (Rule selection strategy). For every state s , agent i , and $r \in V(s, i)$, we say that the rule r matches at state s iff $ant(\bar{r}) \subseteq V(s, i)$ and $cons(\bar{r}) \not\subseteq V(s, i)$. Let $\delta : S \times A_g \rightarrow \mathcal{G}(\mathcal{R})$ be a function that generates matching rule instances of the agent i at state s and $\mathcal{R}_{mat} \subseteq \mathcal{G}(\mathcal{R})$ denotes the set of all matching rule instances of the agent i at state s . A set \mathcal{R}_{sel} is said to be selected rule instances if (i) $\mathcal{R}_{sel} \subseteq \mathcal{R}_{mat}$; and (ii) $\forall \bar{r} \in \mathcal{R}_{sel} \nexists \bar{r}' \in \mathcal{R}_{sel}$ such that $pri(r') \succ pri(r)$.

Now let us denote the set of all possible actions by agent i in a given state s by $T_i(s)$ and its definition is given below:

Definition 5.3 (Available actions). For every state s and agent i ,

1. $Rule_{i,r,\beta} \in T_i(s)$ iff $\bar{r} \in \mathcal{R}_{sel}$, β is a contradictory context (with respect to $cons(\bar{r})$ i.e., if β is α then $cons(\bar{r})$ is $\sim \alpha$ and vice versa) or $\beta \in \Omega$ or if $|V^M(s, i)| = n_M(i)$ then $\beta \in V^M(s, i)$;
2. $Copy_{i,\alpha,\beta} \in T_i(s)$ iff there exists $j \neq i$ such that $\alpha \in V(s, j)$, $\alpha \notin V(s, i)$, $cp_i^{\bar{m}} \in V(s, i)$ for some $m < n_C(i)$, α is of the form $Ask(j, i, P)$ or $Tell(j, i, P)$, and β as before;
3. $Idle_i$ is always in $T_i(s)$.

Definition 5.4 (Effect of actions). For each $i \in A_g$, the result of performing an action act_i in a state $s \in S$ is defined if $act_i \in T_i(s)$ and has the following effect on the assignment of formulas to i in the successor state $s' \in S$:

1. if act_i is $Rule_{i,r,\beta}$: $V(s', i) = V(s, i) \setminus \{\beta\} \cup \{cons(\bar{r})\}$;
2. if act_i is $Copy_{i,\alpha,\beta}$: $cp_i^{\bar{m}} \in V(s, i)$ for some $m \leq n_C(i)$: $V(s', i) = V(s, i) \setminus \{\beta, cp_i^{\bar{m}}\} \cup \{\alpha, cp_i^{\bar{m}+1}\}$;

3. if act_i is *Idle*: $V(s', i) = V(s, i)$.

Now, the definition of the set of models corresponding to a system of rule-based reasoners is given below:

Definition 5.5. $\mathbb{M}(n_M, n_C)$ is the set of models (S, T, V) which satisfies the following conditions:

1. $cp_i^{\bar{0}} \in V(s_0, i)$ where $s_0 \in S$ is the root of (S, T) , $\forall i \in A_g$;
2. $\forall s \in S$ and a tuple of actions $\langle act_1, act_2, \dots, act_{n_{A_g}} \rangle$, if $act_i \in T_i(s)$, $\forall i \in A_g$, then $\exists s' \in S$ such that sTs' and s' satisfies the effects of act_i , $\forall i \in A_g$;
3. $\forall s, s' \in S$, sTs' iff for some tuple of actions $\langle act_1, act_2, \dots, act_{n_{A_g}} \rangle$, $act_i \in T_i(s)$ and the assignment in s' satisfies the effects of act_i , $\forall i \in A_g$;
4. The bound on each agent's memory is set by the following constraint on the mapping V : $|V^M(s, i)| \leq n_M(i)$, and $cp_i^{\bar{n}} \leq n_C(i) \forall s \in S, i \in A_g$.

Note that the bound $n_C(i)$ on each agent i 's communication ability (no branch contains more than $n_C(i)$ *Copy* actions by agent i) follows from the fact that *Copy* _{i} is only enabled if i has performed fewer than $n_C(i)$ copy actions in the past. Below are some abbreviations which will be used in the axiomatization:

- $ByRule_i(P, m) = \neg B_i P \wedge cp_i^{\bar{m}} \wedge \bigvee_{\bar{r} \in \mathfrak{R}_{sel} \wedge cons(\bar{r})=P} (B_i r \wedge \bigwedge_{Q \in ant(\bar{r})} B_i Q)$. This formula describes a state s where it may make a *Rule* transition and believe context P in the next state, m is the value of i 's communication counter, P and Q are ground atomic formulas.
- $ByCopy_i(\alpha, m) = \neg B_i \alpha \wedge B_j \alpha \wedge cp_i^{\bar{m}-1}$, where α is of the form $Ask(j, i, P)$ or $Tell(j, i, P)$, $i, j \in A_g$ and $i \neq j$.

5.5.5 Axiomatization

Now we introduce the axiomatization system.

A1 All axioms and inference rules of CTL^* [Reynolds, 2001].

A2 $\bigwedge_{\alpha \in D_M(i)} B_i \alpha \rightarrow \neg B_i \beta$ for all $D_M(i) \subseteq \Omega$ such that $|D_M(i)| = n_M(i)$ and $\beta \notin D_M(i)$.

This axiom describes that, in a given state, each agent can store maximally at most $n_M(i)$ formulas in its memory,

A3 $\bigvee_{n=0, \dots, n_C(i)} cp_i^{\bar{n}}$, n is value of the communication counter of an agent i corresponding to its *Copy* actions.

A4 $cp_i^{\bar{n}} \rightarrow \neg cp_i^{\bar{m}}$ for any $m \neq n$, which states that at any given time the value of the copy counter of agent i is unique.

A5 $B_i \alpha \rightarrow \neg B_i \sim \alpha$ for any $\alpha \in S_M(i) \cup D_M(i) \subseteq \Omega$,

This axiom states that agent does not believe contradictory contexts,

A6 $B_i r \wedge \bigwedge_{\bar{r} \in \mathcal{R}_{sel} \wedge P \in ant(\bar{r})} B_i P \wedge cp_i^{\bar{n}} \wedge \neg B_i cons(\bar{r}) \rightarrow EX (B_i cons(\bar{r}) \wedge cp_i^{\bar{n}}), i \in A_g$.

This axiom describes that if a rule matches and is selected for execution, its consequent belongs to some successor state.

A7 $cp_i^{\bar{m}} \wedge \neg B_i \alpha \wedge B_j \alpha \rightarrow EX (B_i \alpha \wedge cp_i^{\bar{m}+1})$ where α is of the form $Ask(j, i, P)$ or $Tell(j, i, P)$, $i, j \in A_g, j \neq i, m < n_C(i)$.

This axiom describes transitions made by *Copy* with communication counter increased.

A8 $EX (B_i \alpha \wedge B_i \beta) \rightarrow B_i \alpha \vee B_i \beta$, where α and β are not of the form $Ask(j, i, P)$ and $Tell(j, i, P)$.

This axiom says that at most one new belief is added in the next state.

A9 $B_i\alpha \rightarrow AX B_i\alpha$ for any $\alpha \in S_M(i) \cup \mathfrak{R}$.

This axiom states that an agent $i \in A_g$ always believes formulas residing in its static memory and its rules.

A10 $EX(B_i\alpha \wedge cp_i^{\equiv m}) \rightarrow B_i\alpha \vee ByRule_i(\alpha, m) \vee ByCopy_i(\alpha, m)$ for any $\alpha \in \Omega$.

This axiom says that a new belief can only be added by one of the valid reasoning actions.

A11(a) $start \rightarrow cp_i^{\equiv 0}$ for all $i \in A_g$.

At the start state, the agent has not performed any *Copy* actions.

A11(b) $\neg EX start$.

start holds only at the root of the tree.

A12 $B_i r$ where $r \in \mathfrak{R}$ and $i \in A_g$.

This axiom tells agent i believes its rules.

A13 $\neg B_i r$ where $r \notin \mathfrak{R}$ and $i \in A_g$.

This axiom tells agent i only believes its rules.

A14 $\varphi \rightarrow EX\varphi$, where φ does not contain *start*.

This axiom describes an *Idle* transition by all the agents.

A15 $\bigwedge_{i \in A_g} EX(\bigwedge_{\alpha \in \Gamma_i} B_i\alpha \wedge cp_i^{\equiv m_i}) \rightarrow EX \bigwedge_{i \in A_g} (\bigwedge_{\alpha \in \Gamma_i} B_i\alpha \wedge cp_i^{\equiv m_i})$ for any $\Gamma_i \subseteq \Omega$.

This axiom describes that if each agent i can separately reach a state where it believes formulas in Γ_i , then all agents together can reach a state where for each i , agent i believes formulas in Γ_i .

Let us now define the logic obtained from the above axiomatisation system.

Definition 5.6. $\mathbb{L}(n_M, n_C)$ is the logic defined by the axiomatisation.

5.6 Correctness Proof

Theorem 5.1. $\mathbb{L}(n_M, n_C)$ is sound and complete with respect to $\mathbb{M}(n_M, n_C)$.

5.6.1 Soundness Proof

As the logic \mathcal{L}_{DROCS} is an extension of the \mathcal{L}_{OCRS} , it has similar set of correctness proofs as given in Chapter 4. But \mathcal{L}_{DROCS} has some additional axioms due to non-monotonic reasoning system. In this section we provide the soundness proofs of those axioms only while other proofs are briefly described. The proofs for axioms and rules included in **A1** are given in [Reynolds, 2001]. Axiom **A2** assures that at a state, each agent can store maximally at most $n_M(i)$ formulas in its memory. Axioms **A3** and **A4** force the presence of a unique counter for each agent to record the number of copies it has performed so far. In particular, **A3** makes sure that at least a counter is available for any agent and **A4** guaranties that only one of them is present. In the following, we provide the proof for **A5**, **A6** and **A7**.

Axiom **A5** assures that an agent does not believe on contradictory contexts. Let $\mathbb{M} = (S, T, V) \in \mathbb{M}(n_M, n_C)$, $\pi \in B(S, T)$ and $n \geq 0$. We assume that if $\mathbb{M}, \pi, n \models B_i \alpha$ where $\alpha \in V(\pi_n, i)$, $V(\pi_n, i) = S_M(i) \cup D_M(i)$, and $|V^M(s, i)| \leq n_M(i)$. Then $\sim \alpha$ can not reside at the current state of agent i because the applicable actions of 1 (definition 5.3) ensure that either α or $\sim \alpha$ has to be present in the memory. That is, $\sim \alpha \notin V(\pi_n, i)$. Therefore, $\mathbb{M}, \pi, n \not\models \neg B_i \sim \alpha$.

The proof for axiom **A6** is given as: Let $\mathbb{M} = (S, T, V) \in \mathbb{M}(n_M, n_C)$, $\pi \in B(S, T)$ and $n \geq 0$. We assume that $\mathbb{M}, \pi, n \models B_i r \wedge \bigwedge_{\bar{r} \in \mathfrak{R}_{sel} \wedge P \in ant(\bar{r})} B_i P \wedge cp_i^m \wedge \neg B_i cons(\bar{r})$, for some $r \in \mathfrak{R}$ such that $\bar{r} \in \mathfrak{R}_{sel}$, and $|V^M(s, i)| \leq n_M(i)$. Then $P \in V(\pi_n, i)$ for all $P \in ant(\bar{r})$, and $cons(\bar{r}) \notin V(\pi_n, i)$. This means that the action performed by agent i is $Rule_{i,r,\beta}$. According to the definition of $\mathbb{M}(n_M, n_C)$, $\exists s' \in S$ such that $\pi_n T s'$ and $V(s', i) = V(\pi_n, i) \setminus \{\beta\} \cup \{cons(\bar{r})\}$. Let π' be a branch in $B(S, T)$ such that $\pi'_{\leq n} = \pi_{\leq n}$

and $\pi'_{n+1} = s'$. Then we have $\mathbb{M}, \pi', n+1 \models B_i \text{cons}(\bar{r}) \wedge cp_i^{\bar{m}}$. Therefore, it is obvious that $\mathbb{M}, \pi, n \models EX(B_i \text{cons}(\bar{r}) \wedge cp_i^{\bar{m}})$.

Axiom **A7** is valid by copy action. Let $\mathbb{M} = (S, T, V) \in \mathbb{M}(n_M, n_C)$, $\pi \in B(S, T)$ and $n \geq 0$. We assume that $\mathbb{M}, \pi, n \models cp_i^{\bar{m}} \wedge \neg B_i \alpha \wedge B_j \alpha$, and $|V^M(s, i)| \leq n_M(i)$. Then $cp_i^{\bar{m}} \in V(\pi_n, i)$, $\alpha \notin V(\pi_n, i)$, and $\alpha \in V(\pi_n, j)$, for $i, j \in A_g$, $i \neq j$, and $m < n_C(i)$. This means that the action performed by agent i is $Copy_{i, \alpha, \beta}$. According to the definition of $\mathbb{M}(n_M, n_C)$, $\exists s' \in S \cdot \pi_n T s'$ and $V(s', i) = V(\pi_n, i) \setminus \{\beta, cp_i^{\bar{m}}\} \cup \{\alpha, cp_i^{\bar{m}+1}\}$. Let π' be a branch in $B(S, T)$ such that $\pi'_{\leq n} = \pi_{\leq n}$ and $\pi'_{n+1} = s'$. Then we have $\mathbb{M}, \pi', n+1 \models B_i \alpha \wedge cp_i^{\bar{m}+1}$. Therefore, it is obvious that $\mathbb{M}, \pi, n \models EX(B_i \alpha \wedge cp_i^{\bar{m}+1})$.

5.6.2 Completeness Proof

In completeness, reasoning derives all true statements, which means every true formula is provable. Completeness asserts the existence of rules that allow to deduce every consequence from any set of formula in the logic. For example; $\Omega \models \varphi$ if and only if $\Omega \vdash \varphi$. (If Ω models φ then we can also derive φ from a set of formulas Ω).

Completeness can be shown by constructing a tree model for a consistent formula φ . This is constructed as in the completeness proof introduced in [Reynolds, 2001]. Then we use the axioms to show that this model is in $\mathbb{M}(n_M, n_C)$. Since the initial state of all agents does not restrict the set of formulas they may derive in the future, for simplicity we conjunctively add to φ a tautology that contains all the potentially necessary formulas and message counters, in order to have enough sub-formulas for the construction. We construct a model $\mathbb{M} = (S, T, V)$ for

$$\varphi' = \varphi \wedge \bigwedge_{\alpha \in \Omega} (XB_i \alpha \vee \neg XB_i \alpha) \wedge \bigwedge_{n \in \{0 \dots n_C(i)\}, i \in A_g} (Xcp_i^{\bar{n}} \vee \neg Xcp_i^{\bar{n}})$$

We then prove that \mathbb{M} is in $\mathbb{M}(n_M, n_C)$ by showing that it satisfies all properties listed in Definition 5.5. Axioms **A3** and **A4** show that for any $i \in A_g$, there exists a unique $n \in \{0 \dots n_C\}$ such that at a state s of \mathbb{M} , $cp_i^{\bar{n}} \in V(s, i)$.

For axiom **A5**, we need to prove that for all $s \in S$, $act_i \in T_i(s)$ and $i \in A_g$, there exists a $s' \in S$ such that $s T s'$ and $V(s', i)$ is the resultant state of $V(s, i)$. Axiom **A5** assures that $|V^M(s, i)| \leq n_M(i)$ and the state of agent i can not store conflicting context in the memory. Hence by axioms **A5**, $V(s', i)$ might be different from $V(s, i)$ by overwriting the conflicting context.

We then need to prove that $\forall s \in S, act_i \in T_i(s)$, and $i \in A_g, \exists s' \in S$ such that $s T s'$ and $V(s', i)$ is the result of $V(s, i)$ after i has performed action act_i . Let us consider the case when act_i is $Rule_{i,r,\beta} \in T_i(s)$ for some $r \in \mathfrak{R}$ such that $\bar{r} \in \mathfrak{R}_{sel}$. Since $Rule_{i,r,\beta}$ is applicable at s , $ant(\bar{r}) \subseteq V(s, i)$, $cons(\bar{r}) \notin V(s, i)$. Therefore there exists a MCS χ such that $\chi \supseteq V(s, i)$, and $\bigwedge_{\bar{r} \in \mathfrak{R}_{sel} \wedge P \in ant(\bar{r})} B_i P \wedge cp_i^{\bar{m}} \wedge \neg B_i cons(\bar{r}) \in \chi$, for some $m \in \{0, \dots, n_C\}$ and $|V^M(s, i)| \leq n_M(i)$. By axiom **A6** and Modus Ponens (MP), $EX(B_i cons(\bar{r}) \wedge cp_i^{\bar{m}}) \in \chi$. Therefore, according to the construction, $\exists s' \in S$ such that $s T s'$, $V(s', i) \subseteq \chi'$ for some χ' , and $B_i cons(\bar{r}) \wedge cp_i^{\bar{m}} \in \chi'$. Therefore $V(s', i) = V(s, i) \setminus \{\beta\} \cup \{cons(\bar{r})\}$.

For the proof of axiom **A7**, we need to prove that $\forall s \in S, act_i \in T_i(s)$, and $i \in A_g, \exists s' \in S \cdot s T s'$ and $V(s', i)$ is the result of $V(s, i)$ after i has performed action act_i . Let us consider the case when act_i is $Copy_{i,\alpha,\beta} \in T_i(s)$ for some $r \in \mathfrak{R}$ such that $\bar{r} \in \mathfrak{R}_{sel}$. Since the rule $Copy_{i,\alpha,\beta}$ is applicable at s , $\alpha \notin V(s, i)$ while $\alpha \in V(s, j)$. Therefore, there exists a MCS (Maximal Consistent Set) χ such that $\chi \supseteq V(s, i)$, and $cp_i^{\bar{m}} \wedge \neg B_i \alpha \wedge B_j \alpha \in \chi$, for some $m \in \{0, \dots, n_C(i)\}$ and $|V^M(s, i)| \leq n_M(i)$. By axiom **A7** and MP (Modus Ponens), $EX(B_i \alpha \wedge cp_i^{\bar{m}+1}) \in \chi$. Therefore, according to construction, $\exists s' \in S$ such that $s T s'$, $V(s', i) \subseteq \chi'$ for some χ' , and $B_i \alpha \wedge cp_i^{\bar{m}+1} \in \chi'$. Therefore, $V(s', i) = V(s, i) \setminus \{\beta\} \cup \{\alpha\} \cup \{cp_i^{\bar{m}+1}\}$.

Now we prove that $\forall s, s' \in S \cdot sTs', \exists$ a tuple of actions $\langle act_1, act_2, \dots, act_{n_{Ag}} \rangle$ and $V(s', i)$ is the result of $V(s, i)$ when agent i performs act_i for all $i \in A_g$. By axioms **A8** and **A2**, $V(s', i)$ is different from $V(s, i)$ by at most one formula added and possibly a formula is removed. If no formula is added or removed, we consider act_i to be $Idle_i$.

Let us now consider the case where a formula α is added. By axiom **A10**, if $cp_i^{\bar{m}} \in V(s, i)$ for some $m \in \{0, \dots, n_C\}$ then either $cp_i^{\bar{m}}$ or $cp_i^{m+1} \in V(s', i)$. If $cp_i^{\bar{m}} \in V(s', i)$ then set act_i to be $Rule_{i,r,\beta}$ for some $r \in V(s, i)$ such that $\bar{r} \in \mathfrak{R}_{sel}$, $\alpha = cons(\bar{r}) \notin V(s, i)$. If $cp_i^{\bar{m}+1} \in V(s', i)$ then set act_i to be $Copy_{i,\alpha,\beta}$. Thus, we proved the existence of the tuple $\langle act_1, act_2, \dots, act_{n_{Ag}} \rangle$ for sTs' . Therefore, \mathbb{M} is in $\mathbb{M}(n_M, n_C)$.

At the root s_0 of (S, T) , the construction of the model implies that there exists a maximally consistent set (MCS) χ_0 such that $\chi_0 \supseteq V(s_0, i)$ and $start \in \chi_0$. Therefore, by axiom **A11**, it is trivial that $cp_i^{\bar{0}} \in V(s_0, i)$.

For the $Idle_i \in T_i(s)$ actions, the proof is similar by using MP and axiom **A14**. Then, using axiom **A15** we can show that, for any tuple of actions $\langle act_1, act_2, \dots, act_{n_{Ag}} \rangle$, $act_i \in T_i(s)$ is applicable at $s \in S \forall i \in A_g$, then $\exists s' \in S$ such that $V(s', i)$ is the result of $V(s, i)$ after performing act_i at s by agent i , $\forall i \in A_g$.

5.7 A Simple Health-care Example

To illustrate the use of the proposed framework, let us consider a simple health-care example system consisting of four agents. This system monitors the patient's fever and blood sugar level. Patient care agent receives information from Fever Detector agent and Diabetes Tester agent after certain intervals of time and take appropriate actions. Patient care agent communicate Emergency monitoring agent in case of emergency situation for the patient. We build the ontology for this example system. A fragment of the context modelling ontology of the system is depicted in Fig. 5.2. Figure 5.3 shows an individualised

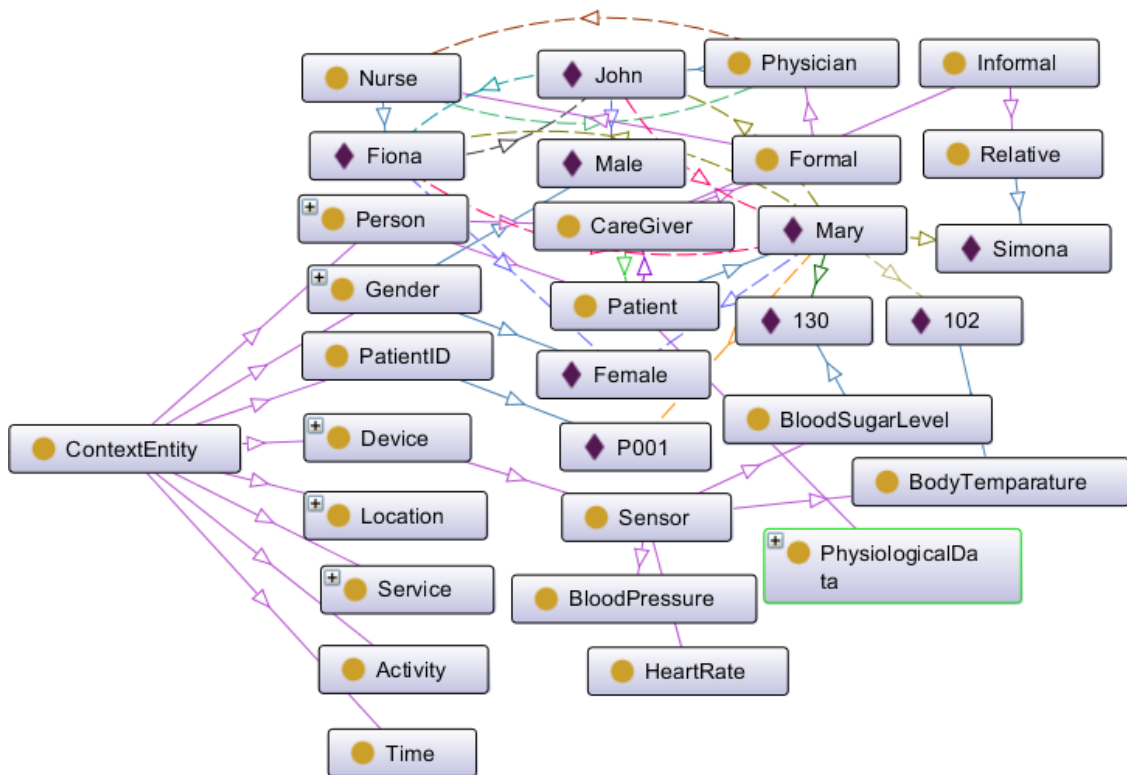


Figure 5.2: A fragment of Home-care Patient's Monitoring System

Property assertions: Mary		
Object property assertions +		
hasBodyTemperature	102	? @ x o
hasCarer	Fiona	? @ x o
hasPatientID	P001	? @ x o
hasCarer	Simona	? @ x o
hasCarer	John	? @ x o
hasGender	Female	? @ x o
hasBloodSugarLevelBeforeMeal	130	? @ x o
Data property assertions +		
hasHeight	160	? @ x o
hasConsciousness	"Yes"^^string	? @ x o
hasName	"Mary Smith"^^Literal	? @ x o
'Tell 2, 1, hasFever'	"High"^^string	? @ x o
hasWeight	60	? @ x o
'Tell 3, 1, hasDBCATEGORY'	"EstablishedDiabetes"^^string	? @ x o
hasAge	65	? @ x o
hasSituation	"Emergency"^^string	? @ x o
Negative object property assertions +		
Negative data property assertions +		
hasSituation	"Emergency"^^string	? @ x o

Figure 5.3: Individualized Patient Ontology

Rules:	
hasDBCategory(?p, "EstablishedDiabetes") -> 'Tell 3, 1, hasDBCategory'(?p, "EstablishedDiabetes")	
'Tell 3, 1, hasDBCategory'(?p, "EstablishedDiabetes") -> hasDBCategory(?p, "EstablishedDiabetes")	
'Tell 1, 4, hasSituation'(?p, "Emergency") -> hasSituation(?p, "Emergency")	
Patient(?p), hasConsciousness(?p, "Yes"), hasFever(?p, "High") -> ~ hasSituation(?p, "Emergency")	
PatientID(?pid), Person(?p), hasPatientID(?p, ?pid) -> Patient(?p)	
Patient(?p), hasSituation(?p, "Emergency") -> 'Tell 1, 4, hasSituation'(?p, "Emergency")	
Patient(?p), hasDBCategory(?p, "EstablishedDiabetes"), hasFever(?p, "High") -> hasSituation(?p, "Emergency")	
BodyTemperature(?temp), Person(?p), hasBodyTemperature(?p, ?temp), greaterThanOrEqual(?temp, 101), lessThanOrEqual(?temp, 103) -> hasFever(?p, "High")	
hasFever(?p, "High") -> 'Tell 2, 1, hasFever'(?p, "High")	
'Tell 2, 1, hasFever'(?p, "High") -> hasFever(?p, "High")	
BloodSugarLevel(?bsl), Person(?p), hasBloodSugarLevelBeforeMeal(?p, ?bsl), greaterThanOrEqual(?bsl, 126) -> hasDBCategory(?p, "EstablishedDiabetes")	

Figure 5.4: Some SWRL Rules

Agent 1: Patient care
Initial facts: Person('Mary'), PatientID('P001'), hasPatientID('Mary', 'P001'), hasConsciousness('Mary', 'Yes')
R11: Person(?p), hasPatientID(?p, ?pid), PatientID(?pid) → Patient(?p)
R12: Tell(2,1, hasFever(?p, 'High')) → hasFever(?p, 'High')
R13: Tell(3,1, hasDBCategory(?p, 'EstablishedDiabetes')) → hasDBCategory(?p, 'EstablishedDiabetes')
R14: Patient(?p), hasFever(?p, 'High'), hasConsciousness(?p, 'Yes') ⇒ ~ hasSituation(?p, 'Emergency')
R15: Patient(?p), hasFever(?p, 'High'), hasDBCategory(?p, 'EstablishedDiabetes') ⇒ hasSituation(?p, 'Emergency')
R16: Patient(?p), hasSituation(?p, 'Emergency') → Tell(1,4, hasSituation(?p, 'Emergency'))
Rule Priority: R15 ≻ R14
Agent 2: Fever detector
Initial facts: Person('Mary'), BodyTemperature('102'), hasBodyTemperature('Mary', '102'), greaterThanOrEqual('102', '101'), lessThanOrEqual('102', '103')
R21: Person(?p), BodyTemperature(?temp), hasBodyTemperature(?p, ?temp), greaterThanOrEqual(?temp, '101'), lessThanOrEqual(?temp, '103') → hasFever(?p, 'High')
R22: hasFever(?p, 'High') → Tell(2,1, hasFever(?p, 'High'))
Agent 3: Diabetes tester
Initial facts: Person('Mary'), BloodSugarLevel('130'), hasBloodSugarLevelBeforeMeal('Mary', '130'), greaterThan('130', '126')
R31: Person(?p), BloodSugarLevel(?bsl), hasBloodSugarLevelBeforeMeal(?p, ?bsl), greaterThan(?bsl, '126') → hasDBCategory(?p, 'EstablishedDiabetes')
R32: hasDBCategory(?p, 'EstablishedDiabetes') → Tell(3,1, hasDBCategory(?p, 'EstablishedDiabetes'))
Agent 4: Emergency
Initial facts:
R41: Tell(1,4, hasSituation(?p, 'Emergency')) → hasSituation(?p, 'Emergency')

Table 5.1: Example Rules for a homecare patients' monitoring context-aware system

patient ontology and Figure 5.4 depicts some SWRL rules. The set of translated rules and initial working memory facts that are distributed to the agents are shown in Table 5.1, and the goal is to infer the formula $B_4 \text{ hasSituation}('Mary', 'Emergency')$ which states that agent 4 believes that the patient Mary has Emergency situation. The reasoning process includes resolving contradictory contextual information.

5.7.1 Verifying System Properties

To verify resource-bounded as well as non-conflicting context properties, one possible run of the system is shown in table 5.2 and 5.3. In the table a newly inferred context at a particular step is shown in blue text. For example, antecedents of rule R11 of agent 1 match the contents of the memory configuration and infers new context $Patient('Mary)$ at step 1. A context which gets overwritten in the next state is shown in red text, and a context which is inferred in the current state and gets overwritten in the next state is shown in magenta text. In the memory configuration, left side of the red vertical bar $|$ represents $S_M(i)$ and its right side represents $D_M(i)$ for each agent i . It shows that the size of $D_M(1)$ is 3 units and the size of $D_M(i)$ is 1 unit for all $2 \leq i \leq 4$. We can observe that the resource requirements for the system to derive the goal formula $B_4 hasSituation('Mary,' Emergency)$ are 3 messages that need to be exchanged by agent 1 and 1 message that needs to be exchanged by each of the other three agents and 10 time steps. One may also observe that, if we reduce the dynamic memory size for agent 1 by 1, then the system will not be able to achieve the desired goal.

We can prove that $X^{10}B_4 hasSituation('Mary,' Emergency)$ (i.e., from the start state, agent 4 believes $hasSituation('Mary,' Emergency)$ in 10 time steps), where X^{10} is the concatenation of ten LTL next operators X .

This is a very simple case; however, if we model a more realistic scenario and increase the problem size, the verification task would be hard to do by hand. Therefore it is more convenient to use an automatic method to verify them, for example using model checking techniques. In Chapter 6, we show how a \mathcal{L}_{DROCS} model can be encoded using a standard model checker such as for example the Maude LTL model checker [Eker et al., 2003] and its interesting properties can be verified.

#Steps	Patient care		Fever detector		Diabetes tester		Emergency	
	Memory Config.1	Action1 # Msg1	Memory Config.2	Action2 #Msg2	Memory Config.3	Action3 #Msg3	Memory Config.4	Action4 #Msg4
0	{Person('Mary), PatientID('P001), hasPatientID('Mary,' P001), hasConsciousness('Mary,' Yes) -, -, - }	-	{Person('Mary), BodyTemperature('102), hasBodyTemperature('Mary,' 102), greaterThanOrEqual('102,' 101), lessThanOrEqual('102,' 103) -, -, - }	-	{Person('Mary), BloodSugarLevel('130), hasBloodSugarLevelBeforeMeal('Mary,' 130), greaterThan('130,' 126) -, -, - }	-	{-}	-
1	{Person('Mary), PatientID('P001), hasPatientID('Mary,' P001), hasConsciousness('Mary,' Yes) Patient('Mary), -, - }	Rule (R11)	{Person('Mary), BodyTemperature('102), hasBodyTemperature('Mary,' 102), greaterThanOrEqual('102,' 101), lessThanOrEqual('102,' 103) hasFever('Mary,' High) }	Rule (R21)	{Person('Mary), BloodSugarLevel('130), hasBloodSugarLevelBeforeMeal('Mary,' 130), greaterThan('130,' 126) hasDBCategory('Mary,' EstablishedDiabetes) }	Rule (R31)	{-}	Idle
2	{Person('Mary), PatientID('P001), hasPatientID('Mary,' P001), hasConsciousness('Mary,' Yes) Patient('Mary), -, - }	Idle	{Person('Mary), BodyTemperature('102), hasBodyTemperature('Mary,' 102), greaterThanOrEqual('102,' 101), lessThanOrEqual('102,' 103) Tell(2, 1, hasFever('Mary,' High)) }	Rule (R22)	{Person('Mary), BloodSugarLevel('130), hasBloodSugarLevelBeforeMeal('Mary,' 130), greaterThan('130,' 126) Tell(3, 1, hasDBCategory('Mary,' EstablishedDiabetes)) }	Rule (R32)	{-}	Idle
3	{Person('Mary), PatientID('P001), hasPatientID('Mary,' P001), hasConsciousness('Mary,' Yes) Patient('Mary), Tell(2, 1, hasFever('Mary,' High), - }	Copy	{Person('Mary), BodyTemperature('102), hasBodyTemperature('Mary,' 102), greaterThanOrEqual('102,' 101), lessThanOrEqual('102,' 103) Tell(2, 1, hasFever('Mary,' High)) }	Idle	{Person('Mary), BloodSugarLevel('130), hasBloodSugarLevelBeforeMeal('Mary,' 130), greaterThan('130,' 126) Tell(3, 1, hasDBCategory('Mary,' EstablishedDiabetes)) }	Idle	{-}	Idle
4	{Person('Mary), PatientID('P001), hasPatientID('Mary,' P001), hasConsciousness('Mary,' Yes) Patient('Mary), Tell(2, 1, hasFever('Mary,' High), - }	Copy	{Person('Mary), BodyTemperature('102), hasBodyTemperature('Mary,' 102), greaterThanOrEqual('102,' 101), lessThanOrEqual('102,' 103) Tell(2, 1, hasFever('Mary,' High)) }	Idle	{Person('Mary), BloodSugarLevel('130), hasBloodSugarLevelBeforeMeal('Mary,' 130), greaterThan('130,' 126) Tell(3, 1, hasDBCategory('Mary,' EstablishedDiabetes)) }	Idle	{-}	Idle
5	{Person('Mary), PatientID('P001), hasPatientID('Mary,' P001), hasConsciousness('Mary,' Yes) Patient('Mary), hasFever('Mary,' High), Tell(3, 1, hasDBCategory('Mary,' EstablishedDiabetes)) }	Rule (R12)	{Person('Mary), BodyTemperature('102), hasBodyTemperature('Mary,' 102), greaterThanOrEqual('102,' 101), lessThanOrEqual('102,' 103) Tell(2, 1, hasFever('Mary,' High)) }	Idle	{Person('Mary), BloodSugarLevel('130), hasBloodSugarLevelBeforeMeal('Mary,' 130), greaterThan('130,' 126) Tell(3, 1, hasDBCategory('Mary,' EstablishedDiabetes)) }	Idle	{-}	Idle

Table 5.2: One Possible Run (Reasoning) of the System (Continued on Table 5.3)

#Steps	Patient care			Fever detector			Diabetes tester			Emergency	
	Memory Config.1	Action1	# Msg1	Memory Config.2	Action2	#Msg2	Memory Config.3	Action3	#Msg3	Memory Config.4	Action4
6	{Person('Mary'), PatientID('P001'), hasPatientID('Mary', P001), hasConsciousness('Mary', Yes) Patient('Mary'), hasFever('Mary', High), hasDBCategory('Mary', EstablishedDiabetes)}	Rule (R13)	2	{Person('Mary'), BodyTemperature('102'), hasBodyTemperature('Mary', '102'), greaterThanOrEqual('102', '101'), lessThanOrEqual('102', '103') Tell(2, 1, hasFever('Mary', High)))}	Idle	1	{Person('Mary'), BloodSugarLevel('130'), hasBloodSugarLevelBeforeMeal('Mary', '130'), greaterThan('130', '126') Tell(3, 1, hasDBCategory('Mary', EstablishedDiabetes))}	Idle	1	{-}	Idle
7	{Person('Mary'), PatientID('P001'), hasPatientID('Mary', P001), hasConsciousness('Mary', Yes) Patient('Mary'), hasSituation('Mary', 'Emergency'), hasDBCategory('Mary', EstablishedDiabetes)}	Rule (R15-R14 Resolving conflicting context)	2	{Person('Mary'), BodyTemperature('102'), hasBodyTemperature('Mary', '102'), greaterThanOrEqual('102', '101'), lessThanOrEqual('102', '103') Tell(2, 1, hasFever('Mary', High)))}	Idle	1	{Person('Mary'), BloodSugarLevel('130'), hasBloodSugarLevelBeforeMeal('Mary', '130'), greaterThan('130', '126') Tell(3, 1, hasDBCategory('Mary', EstablishedDiabetes))}	Idle	1	{-}	Idle
8	{Person('Mary'), PatientID('P001'), hasPatientID('Mary', P001), hasConsciousness('Mary', Yes) Patient('Mary'), hasSituation('Mary', 'Emergency'), Tell(1, 4, hasSituation('Mary', 'Emergency'))}	Rule (R16)	3	{Person('Mary'), BodyTemperature('102'), hasBodyTemperature('Mary', '102'), greaterThanOrEqual('102', '101'), lessThanOrEqual('102', '103') Tell(2, 1, hasFever('Mary', High)))}	Idle	1	{Person('Mary'), BloodSugarLevel('130'), hasBloodSugarLevelBeforeMeal('Mary', '130'), greaterThan('130', '126') Tell(3, 1, hasDBCategory('Mary', EstablishedDiabetes))}	Idle	1	{-}	Idle
9	{Person('Mary'), PatientID('P001'), hasPatientID('Mary', P001), hasConsciousness('Mary', Yes) Patient('Mary'), hasSituation('Mary', 'Emergency'), Tell(1, 4, hasSituation('Mary', 'Emergency'))}	Idle	3	{Person('Mary'), BodyTemperature('102'), hasBodyTemperature('Mary', '102'), greaterThanOrEqual('102', '101'), lessThanOrEqual('102', '103') Tell(2, 1, hasFever('Mary', High)))}	Idle	1	{Person('Mary'), BloodSugarLevel('130'), hasBloodSugarLevelBeforeMeal('Mary', '130'), greaterThan('130', '126') Tell(3, 1, hasDBCategory('Mary', EstablishedDiabetes))}	Idle	1	{Tell(1, 4, hasSituation('Mary', 'Emergency'))}	Copy
10	{Person('Mary'), PatientID('P001'), hasPatientID('Mary', P001), hasConsciousness('Mary', Yes) Patient('Mary'), hasSituation('Mary', 'Emergency'), Tell(1, 4, hasSituation('Mary', 'Emergency'))}	Idle	3	{Person('Mary'), BodyTemperature('102'), hasBodyTemperature('Mary', '102'), greaterThanOrEqual('102', '101'), lessThanOrEqual('102', '103') Tell(2, 1, hasFever('Mary', High)))}	Idle	1	{Person('Mary'), BloodSugarLevel('130'), hasBloodSugarLevelBeforeMeal('Mary', '130'), greaterThan('130', '126') Tell(3, 1, hasDBCategory('Mary', EstablishedDiabetes))}	Idle	1	{hasSituation('Mary', 'Emergency')}	Infer (R41)

Table 5.3: One Possible Run (Reasoning) of the System

5.8 Conclusion

In this chapter, we proposed a logical framework for modelling context-aware systems as multi-agent non-monotonic rule-based agents, and the resulting logic \mathcal{L}_{DROCS} allows us to describe a set of ontology-driven rule-based non-monotonic reasoning agents with bounds on time, memory, and communication. Agents use defeasible reasoning technique to reason with inconsistent information. The proposed framework allows us to determine how much time (measured as rule-firing cycles) are required to generate certain contexts, how many messages must be exchanged among agents, and how much space (memory) is required for an agent for the reasoning.

In the next chapter, we discuss how ontological knowledge can be translated into Horn-clause rules. We present an OWL-API based *Onto-HCR* translator whose task is to extract the ontology axioms and then translate them into a plain text of Horn-clause rules. We also show how to encode a \mathcal{L}_{DROCS} model considering a smart environment case study and verify its interesting resource-bounded properties as well as non-conflicting contextual properties automatically.

Chapter 6

Ontology-based System Modelling and Verification

6.1 Chapter Objectives

- To model context-aware systems based on ontologies.
- To translate ontologies to a set of Horn-clause rules.
- To verify an example system using Maude LTL model checker.

6.2 Motivation

Following the theoretical logical frameworks discussed in the previous chapters, we have realized the significance and need to look at practical aspects of the system. In this chapter, we initially describe how ontological knowledge can be translated into Horn-clause rules. For experiment purposes, we develop a tool *Onto-HCR* to translate ontology axioms into Horn-clause rules. To practically model the system, we develop a smart environment case study whose rules are derived from an ontology. For this purpose, the smart environment case study has been developed using Protégé [Protégé, 2011] ontology editor. To verify the correctness of the system, we describe how a $\mathcal{L}_{\mathcal{DROCS}}$ model can be encoded using the

Maude LTL model checker [Eker et al., 2003] and its interesting properties can be verified automatically.

6.3 Translation of an ontology into a set of Rules

This section describes the theoretical approach of how ontologies can be translated into a set of Horn-clause rules. In the literature, several studies have focused on the translation of ontology axioms to description logic (DL) and DL knowledge-base to Defeasible Logic Programming (DeLP) [Faruqui, 2012, Grosz et al., 2003, Gómez et al., 2006, Gómez et al., 2007]. Our approach is driven by insight understanding the DL ontology and the mapping between DL ontology and DeLP. In the following sections, we discuss the step-by-step process of how ontological knowledge can be translated into Horn-clause rules format including non-monotonic rules.

6.3.1 Translating Ontology Axioms into DL Knowledge-base

OWL 2 RL Axioms	DL Syntax	Horn-Clause Rules
SubClassOf	$C \sqsubseteq D$	$C(a) \rightarrow D(a)$
EquivalentClassOf	$C \equiv D$	$\{C(a) \rightarrow D(a), C(a) \leftarrow D(a)\}$
SubObjectPropertyOf	$R \sqsubseteq S$	$R(a, b) \rightarrow S(a, b)$
ObjectPropertyChain	$R \circ S \sqsubseteq T$	$\{R(a, b) \wedge S(b, c)\} \rightarrow T(a, c)$
EquivalentObjectPropertyOf	$R \equiv S$	$\{R(a, b) \rightarrow S(a, b), R(a, b) \leftarrow S(a, b)\}$
InverseObjectPropertyOf	$R \equiv S^{-}$	$\{R(a, b) \rightarrow S(b, a), R(a, b) \leftarrow S(b, a)\}$
ObjectPropertyDomain	$\top \sqsubseteq \forall R^{-}.C$	$R(a, b) \rightarrow C(a)$
ObjectPropertyRange	$\top \sqsubseteq \forall R.C$	$R(a, b) \rightarrow C(b)$
SymmetricObjectProperty	$R \equiv R^{-}$	$R(a, b) \rightarrow R(b, a)$
TransitiveObjectProperty	$R \circ R \sqsubseteq R$	$\{R(a, b) \wedge R(b, c)\} \rightarrow R(a, c)$
ObjectUnionOf	$C_1 \sqcup C_2 \sqsubseteq D$	$\{C_1(a) \rightarrow D(a), C_2(a) \rightarrow D(a)\}$
ObjectIntersectionOf	$C_1 \sqcap C_2 \sqsubseteq D$	$C_1(a) \wedge C_2(a) \rightarrow D(a)$
ObjectAllValuesFrom	$C \sqsubseteq \forall R.D$	$C(a) \wedge R(a, b) \rightarrow D(b)$
ObjectSomeValuesFrom	$\exists R.C \sqsubseteq D$	$R(a, b) \wedge C(b) \rightarrow D(a)$

Table 6.1: Translating OWL 2 RL Axioms into DL Knowledge-base

As we have already discussed in Chapter 3, DL is a decidable fragment of First Order logic (FOL). Logic program (LP) is also closely related to Horn fragment of FOL but neither

included by nor includes FOL [Baral and Gelfond, 1994]. For example, disjunction can be expressible in FOL but is not expressible in LP. On the other hand, logic programming has several expressible features which are used in rule-based applications but not expressible in FOL. Negation as Failure (NaF) could be one of the examples of LP expressing a kind of logical non-monotonicity. Description Logic Program (DLP) is an intermediate Knowledge representation to establish the correlation between DL and Logic programming. DLP is considered as a subset of DL as LP includes non-monotonicity which can be viewed as ontology sub-language. The question arises here why we need DLP. Because we need to model the domain in OWL 2 RL ontology for our logical frameworks and OWL 2 RL design was influenced by DLP [Grosz et al., 2003]. This is certainly an interesting scenario, as DL is based on ontology and is a subset of FOL. So, ontology (OWL 2 RL) axioms can be mapped to their corresponding DL axioms and then DL axioms can be directly translated into Horn-clause rules. Table 6.1 shows the translation from OWL 2 RL axioms to DL axioms which is then translated to a set of Horn-clause rules. These rules are used to develop context-aware agents. Here we show how to translate DL axioms into a set of non-monotonic Horn-clause rules for the logical framework \mathcal{L}_{DROCS} . The Onto-HCR translator, given in Section 6.4.2, extracts these OWL 2 RL axioms from an ontology and then translate them into a set of plain text Horn-clause rules.

6.3.2 Translating DL Knowledge-base into Defeasible Logic Programming (DeLP)

In this section, we provide the translation from DL knowledge-base to DeLP. DeLP is a language which combines LP with defeasible argumentation for knowledge representation and reasoning to decide between contradictory conclusions based on certain evidence [García and Simari, 2004]. DL ontology knowledge-base can be translated into an equivalent defeasible logic program. The reason behind particular logic model of the system

is inconsistent ontologies. The traditional reasoners such as Pellet [Sirin et al., 2007], Racer [Haarslev and Möller, 2001], etc. issue error message due to inconsistent ontology while performing reasoning and halt further processing. We followed similar approach proposed by [Gómez et al., 2010, Gómez et al., 2007]. So, DL classes and properties can be translated to their corresponding DeLP axioms. As mentioned in [Gómez et al., 2010], DL sentences can be mapped to DeLP strict and defeasible rules.

We define a set of strict and defeasible rules which are derived from an OWL 2 RL ontology \mathcal{O} by $\mathcal{P} = (\mathcal{R}^s, \mathcal{R}^d)$. The elements of \mathcal{R}^s are of the form $P_1, P_2, \dots, P_n \rightarrow P$ and elements of \mathcal{R}^d are of the form $P_1, P_2, \dots, P_n \Rightarrow P$. A DL knowledge base $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ has two components, where \mathcal{T} represents terminology box (TBox) and \mathcal{A} represents assertion box (ABox). For each set of strict and defeasible rules, TBox \mathcal{T} is further segregated into two disjoint sets, namely a strict terminology \mathcal{T}_s and a defeasible terminology \mathcal{T}_d . Intuitively, the set of strict rules \mathcal{R}^s in \mathcal{P} corresponds to the ABox \mathcal{A} joined with \mathcal{T}_s in \mathcal{KB} . In the same way, the set of defeasible rules \mathcal{R}^d corresponds to the ABox \mathcal{A} joined with \mathcal{T}_d in \mathcal{KB} . Mapping From DL axioms to strict and defeasible terminologies is given in Table 6.2. For Table 6.2, some rules are extracted from smart environment example scenario given in Section 6.5

6.3.3 Translating Strict and Defeasible Terminologies to Horn-clause Rules

We use translation functions to translate strict and defeasible terminologies into a set of rules. Let $f_s : \mathcal{T}_s \cup \mathcal{A} \rightarrow \mathcal{R}^s$ be a translation function that translates a set of strict TBox axioms into strict rules, and $f_d : \mathcal{T}_d \cup \mathcal{A} \rightarrow \mathcal{R}^d$ be a translation function that translates a set of defeasible TBox axioms into defeasible rules. Table 6.3 shows the translation from strict and defeasible rule terminologies into a set of Horn-clause rules.

DL Knowledge Base (\mathcal{KB})
Strict Rules Terminology (TBox) \mathcal{T}_s <i>Rule1</i> : $Person \sqcap \exists hasPatientID.PatientID \sqsubseteq Patient$ <i>Rule2</i> : $Ambulance \sqcap \exists hasAmbulanceCallFor.Ambulance \sqsubseteq isRescuedBy.Ambulance$ <i>Rule3</i> : $\exists hasSituation.Emergency \sqcap \exists hasGPSLocation.GPS \sqsubseteq hasWarningSignAt.GPS$ <i>Rule4</i> : $Person \sqcap \exists hasSystolicBP.SystolicBP \sqcap \exists hasDiastolicBP.DiastolicBP \sqcap$ $\exists hasSystolicBP. \geq 160 \sqcap \exists hasDiastolicBP. \geq 100 \sqsubseteq hasBloodPressure.Stage2hypertention$
Defeasible Rules Terminology (TBox) \mathcal{T}_d <i>Rule5</i> : $Patient \sqcap \exists hasBloodPressure.Normal \sqsubseteq \sim hasSituation.Emergency$ <i>Rule6</i> : $Patient \sqcap \exists hasBloodPressure.Stage2hypertention \sqsubseteq hasSituation.Emergency$ <i>Rule7</i> : $Patient \sqcap \exists hasFever.Normal \sqcap \exists hasDBCategory.Type2Diabetes \sqsubseteq hasSituation.OnCall$ <i>Rule8</i> : $Person \sqcap MotionDetector \sqsubseteq hasOccupancy.Yes$
Communication Rules <i>Rule9</i> : $Patient \sqcap \exists hasSituation.Emergency \sqsubseteq Tell1To7hasSituation.Emergency$ <i>Rule10</i> : $\exists Tell10To6hasAmbulanceCallFor.GPS \sqsubseteq hasAmbulanceCallFor.GPS$ <i>Rule11</i> : $\exists Tell1To7hasSituation.Emergency \sqsubseteq hasSituation.Emergency$ <i>Rule12</i> : $\exists Tell9To7hasGPSLocation.GPS \sqsubseteq hasGPSLocation.GPS$ <i>Rule13</i> : $\exists hasWarningSignAt(?p, ?loc) \sqsubseteq Tell7To10hasWarningSignAt.GPS$
Assertional Box (ABox): \mathcal{A} <i>Philip</i> : <i>Person</i> <i>P001</i> : <i>PatientID</i> <i>Philip</i> : <i>Patient</i> <i>KajangTownVan3</i> : <i>Ambulance</i> <i>Yes</i> : <i>MotionDetector</i> <i>165</i> : <i>SystolicBP</i> <i>105</i> : <i>DiastolicBP</i> $\langle Philip, P001 \rangle : hasPatientID$ $\langle Philip, KFCKajangTown \rangle : hasAmbulanceCallFor$ $\langle Philip, KajangTownVan3 \rangle : isRescuedBy$ $\langle Philip, Emergency \rangle : hasSituation$ $\langle Philip, KFCKajangTown \rangle : hasGPSLocation$ $\langle Philip, KFCKajangTown \rangle : hasWarningSignAt$ $\langle Philip, 165 \rangle : hasSystolicBP$ $\langle Philip, 105 \rangle : hasDiastolicBP$ $\langle 165, 160 \rangle : greaterThan$ $\langle 105, 100 \rangle : greaterThan$ $\langle Philip, OnCall \rangle : hasSituation$ $\langle Philip, KFCKajangTown \rangle : hasAlarmFor$ $\langle Philip, KFCKajangTown \rangle : hasGPSLocation$ $\langle Philip, Normal \rangle : hasBloodPressure$ $\langle Philip, Emergency \rangle : \sim hasSituation$ $\langle Philip, Stage2hypertention \rangle : hasBloodPressure$ $\langle Philip, Normal \rangle : hasFever$ $\langle Philip, Type2Diabetes \rangle : hasDBCategory$ $\langle Philip, Yes \rangle : hasOccupancy$

Table 6.2: Mapping from DL Axioms to Strict and Defeasible Terminologies

Translating strict and defeasible terminologies to Rules
Translation from TBox axioms \mathcal{T}_s to Strict Rules <p> <i>Rule1</i> : $Person(?p), hasPatientID(?p, ?pid), PatientID(?pid) \rightarrow Patient(?p)$ <i>Rule2</i> : $Ambulance(?amb), hasAmbulanceCallFor(?p, ?loc) \rightarrow isRescuedBy(?p, ?amb)$ <i>Rule3</i> : $hasSituation(?p, "Emergency"), hasGPSLocation(?p, ?loc) \rightarrow hasWarningSignAt(?p, ?loc)$ <i>Rule4</i> : $Person(?p), SystolicBP(?sbp), DiastolicBP(?dbp), hasSystolicBP(?p, ?sbp),$ $hasDiastolicBP(?p, ?dbp), greaterThan(?sbp, 160), greaterThan(?dbp, 100)$ $\rightarrow hasBloodPressure(?p, "Stage2hypertention")$ </p> Translation from TBox axioms \mathcal{T}_d to Defeasible Rules <p> <i>Rule5</i> : $Patient(?p), hasBloodPressure(?p, "Normal") \Rightarrow \sim hasSituation(?p, "Emergency")$ <i>Rule6</i> : $Patient(?p), hasBloodPressure(?p, "Stage2hypertention") \Rightarrow hasSituation(?p, "Emergency")$ <i>Rule7</i> : $Patient(?p), hasFever(?p, "Normal"), hasDBCategory(?p, "Type2Diabetes")$ $\Rightarrow hasSituation(?p, "OnCall")$ <i>Rule8</i> : $Person(?p), MotionDetector('Yes') \Rightarrow hasOccupancy(?p, "Yes")$ </p> Communication Rules <p> <i>Rule9</i> : $Patient(?p), hasSituation(?p, "Emergency") \rightarrow Tell(1, 7, hasSituation(?p, "Emergency"))$ <i>Rule10</i> : $Tell(10, 6, hasAmbulanceCallFor(?p, ?loc)) \rightarrow hasAmbulanceCallFor(?p, ?loc)$ <i>Rule11</i> : $Tell(1, 7, hasSituation(?p, "Emergency")) \rightarrow hasSituation(?p, "Emergency")$ <i>Rule12</i> : $Tell(9, 7, hasGPSLocation(?p, ?loc)) \rightarrow hasGPSLocation(?p, ?loc)$ <i>Rule13</i> : $hasWarningSignAt(?p, ?loc) \rightarrow Tell(7, 10, hasWarningSignAt(?p, ?loc))$ </p>

Table 6.3: Mapping from Strict and Defeasible Terminologies into Rules

Based on the work presented by [Gómez et al., 2010, Gómez et al., 2007, Grosz et al., 2003], a subset of description logic languages can be effectively mapped to Horn-clause logics and DL axioms correspond to first order rules which can be transformed into Horn clause rules. These rules are of the form of conjunctive concepts and roles which can be directly expressed in the body of the DeLP rules.

6.4 Onto-HCR Translator

We developed a translator to translate ontology axioms into Horn-clause rules. This translation process is automated which uses Java-based translator and uses OWL API based framework. We shortly survey some preliminary concepts first and then discuss the Onto-HCR translator's functionalities in order to understand the clear picture of the system.

6.4.1 OWL API

The OWL API is a high level application programming interface (API) that supports the creation and manipulation of OWL ontologies. OWL API was initially introduced in 2003 [Bechhofer et al., 2003] and then later significant changes have been made in order to ensure the correct design patterns suitable with the OWL 2 specifications. Since its initial development, it has been used in various development projects such as Protégé 4 [Knublauch et al., 2004], Pellet reasoner [Sirin et al., 2007], NeOn Toolkit [Haase et al., 2008] and OntoTrack [Liebig and Noppens, 2004] etc. The flexible design pattern of OWL API allows third parties to develop or customize alternative implementations for their components. It is a Java based application programming interface for loading, saving, parsing and serializing ontologies in many different syntaxes (defined in W3C specifications), for example, OWL/XML, RDF/XML, functional syntax, Manchester syntax, KRSS, Turtle syntax, etc. OWL API has a set of interfaces for probing, manipulating and reasoning with OWL ontologies. The main features of OWL API are axiom-centric abstraction, reasoner interfaces, validations for different OWL 2 profiles and first class change support. The OWL API has very close association with the OWL 2 structural specification. The reference implementation of OWL API encompassess validators for all three OWL 2 profiles [Horridge and Bechhofer, 2009, Horridge and Bechhofer, 2011].

6.4.1.1 OWL API Design

The design of the OWL API corresponds to the OWL 2 Structural Specification and this dynamic design model allows developers to provide flexible implementations for major components of the system. The recent developments in OWL API design has effectively filled the gap and meet the needs for OWL ontology based applications, reasoners and editors. As we see in Figure 6.1 which is taken from [Horridge and Bechhofer, 2009], the ontology is viewed as a set of axioms and annotation. In OWL API, the names and hierarchies for the axioms, class expressions and entities correspond to the OWL structural specification. Indeed, there is a proximal one to one translation between OWL API model

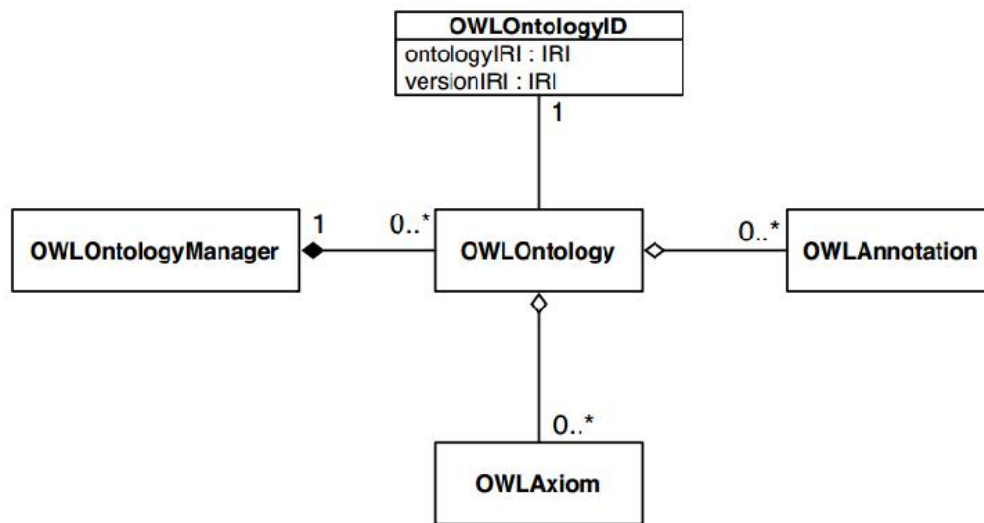


Figure 6.1: OWL API Model

interfaces and the OWL 2 Structural Specification, implying that this becomes easier to correlate the high level OWL 2 specification with the design of the API [Horridge and Bechhofer, 2009].

OWL API's model has a number of interfaces and classes to represent the ontology axioms, for example, OWLAxioms. These are read-only interfaces to access OWLAxioms. Some of the major interfaces are listed below:

- OWLOntology provides access to the axioms which are contained in the ontology.
- OWLOntologyManager is an instance of OWLOntology interface, which acts as a key role for creating, loading, saving and changing ontologies. Each instance of an ontology has a unique identity in a particular ontology manager. Without ontology manager, the ontology can not be created or loaded. Any change made to the ontology is only done by the ontology manager.
- IRI (International Resource Identifier) loads the ontology from the web using its unique identity. OWLEntities such as class names, data and object properties and named individuals can be identified with an IRI.

- OWLAnnotation are used in different types of annotation axioms to bind annotations to their subjects. It extends OWLObject to represent annotated axioms.
- OWL Axiom is a public interface that extends the OWLObject to represent entities and their corresponding relationships.

6.4.1.2 Ontology Management

The OWLOntology interface is a central point to access axioms efficiently from ontologies. Diverse usage of the OWLOntology interface produce distinctive storage structure in the ontologies. The OWLOntologyManager is an instance of OWLOntology interface that acts as central hub for creating, loading, saving and updating ontologies via its manager. Each instance of an ontology is created and manipulated by its own particular OWLOntologymanager. This kind of system design provides a centralized mechanism for the client applications to be monitored and controlled by one central access point [Horridge and Bechhofer, 2009].

A key advantage of conjoining OWL 2 structural specification with OWL API is its free syntax style which means that there is no obligation to any specific syntax. By default, RDF/XML is the only one syntax supported in OWL implementation. However, there are several other syntaxes which are optimized for different purposes. For example, OWL/XML syntax allows ontologies to store in plain XML format, Turtle syntax specifies RDF serialization while Manchester syntax postulate the human readable serialization for OWL ontologies. OWL API does not have a direct support for reading and writing ontologies in different syntaxes. The uses of parser and renderer in the reference implementation of OWL API make this task easier to customize ontologies in different syntaxes. When a specific parser is selected, ontologies are loaded and saved back in the same format from which it is parsed. The OWL API has a programming interface to manage ontology profiles. The ontology profiles validation and its imports closure are performed by OWL API validation frameworks. Reasoning for these profile could be executed by a rule engine.

6.4.2 The Onto-HCR's Main Features

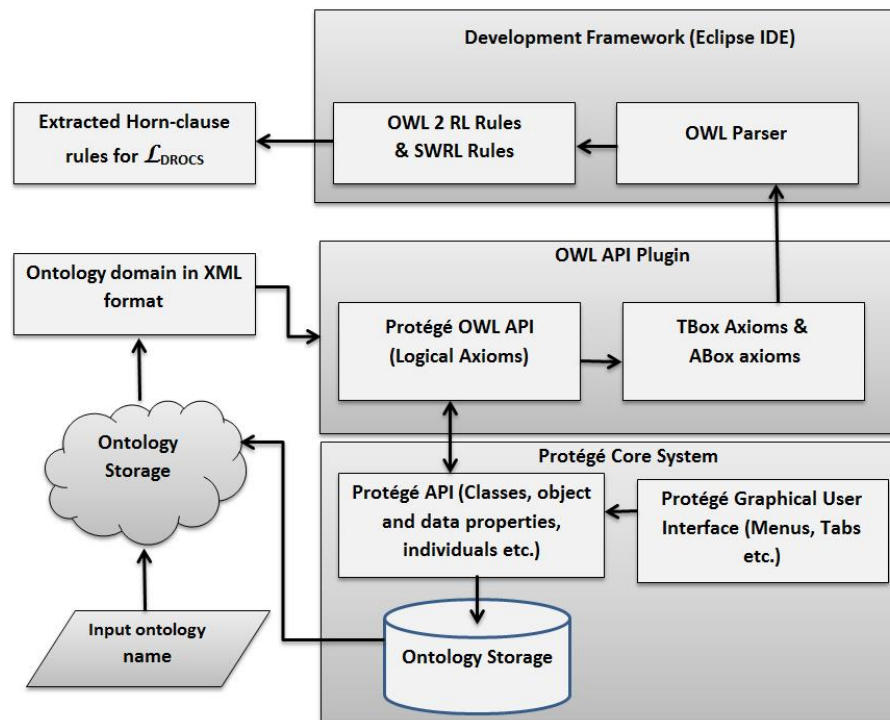


Figure 6.2: Onto-HCR Flow Chart

```

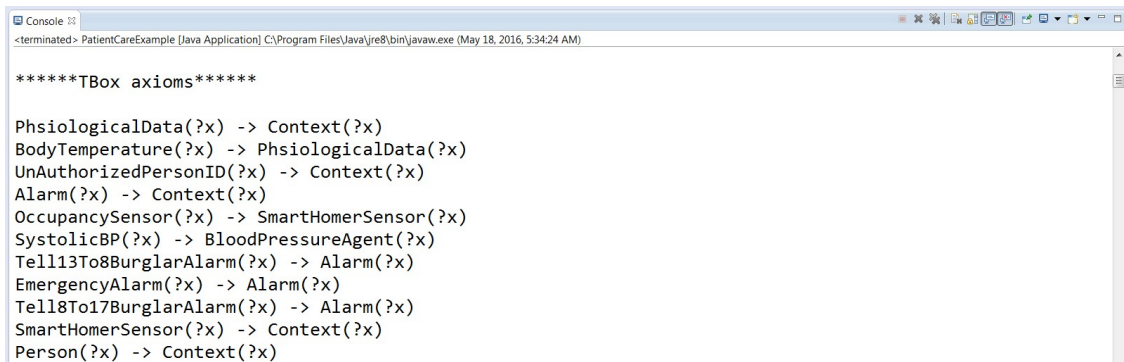
Console 33
<terminated> PatientCareExample [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (May 18, 2016, 5:34:24 AM)

Welcome to Onto-HCR Translator
*****

List of ontologies
=====
1. Smart Patient Care System
2. Home Care System
3. A simple example
4. Patient Care System
5. Smart Environment Ontology
Please enter your choice : 5
  
```

Figure 6.3: Main Menu

We developed a Java-based translator using OWL API version 3.4.10. We chose Eclipse development framework to translate ontology axioms (which are extracted from the published ontology) into a set of plain Horn-clause rules. Each ontology has an ontology IRI (International Resource Identifier) to identify ontology and their classes, properties and individuals. We consider ontology IRI to access the elements from the ontology. The



```

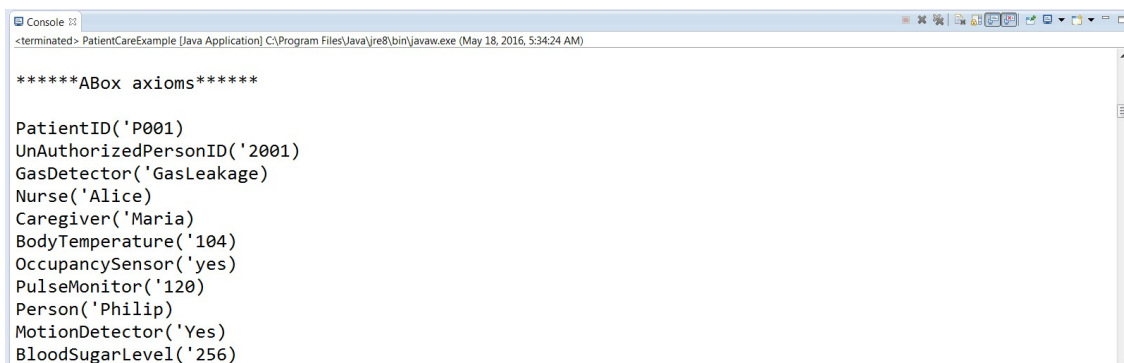
Console
<terminated> PatientCareExample [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (May 18, 2016, 5:34:24 AM)

*****TBox axioms*****

PhysiologicalData(?x) -> Context(?x)
BodyTemperature(?x) -> PhysiologicalData(?x)
UnauthorizedPersonID(?x) -> Context(?x)
Alarm(?x) -> Context(?x)
OccupancySensor(?x) -> SmartHomerSensor(?x)
SystolicBP(?x) -> BloodPressureAgent(?x)
Tell113To8BurglarAlarm(?x) -> Alarm(?x)
EmergencyAlarm(?x) -> Alarm(?x)
Tell18To17BurglarAlarm(?x) -> Alarm(?x)
SmartHomerSensor(?x) -> Context(?x)
Person(?x) -> Context(?x)

```

Figure 6.4: Some of the TBox Axioms



```

Console
<terminated> PatientCareExample [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (May 18, 2016, 5:34:24 AM)

*****ABox axioms*****

PatientID('P001)
UnauthorizedPersonID('2001)
GasDetector('GasLeakage)
Nurse('Alice)
Caregiver('Maria)
BodyTemperature('104)
OccupancySensor('yes)
PulseMonitor('120)
Person('Philip)
MotionDetector('Yes)
BloodSugarLevel('256)

```

Figure 6.5: Some of the ABox Concepts Axioms



```

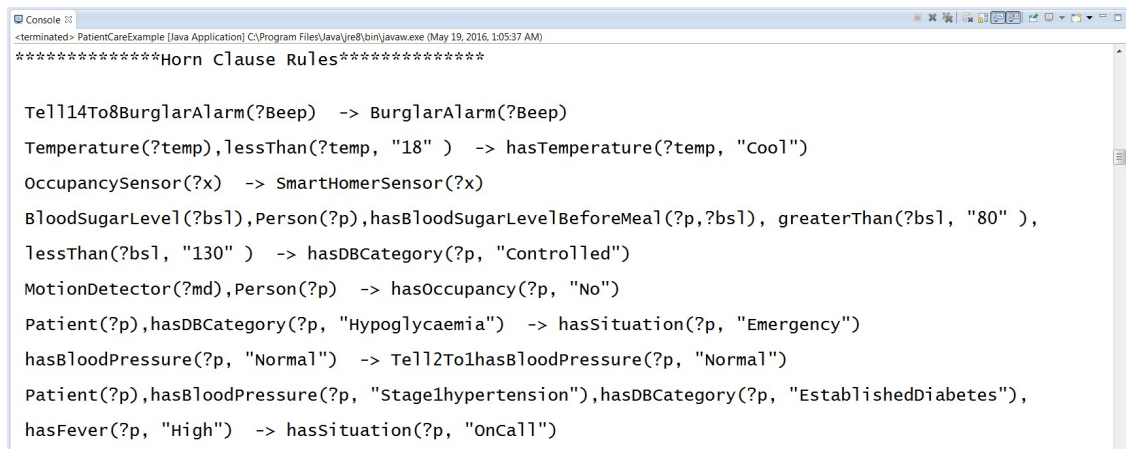
Console
<terminated> PatientCareExample [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (May 19, 2016, 1:05:37 AM)

*****ABox axioms*****

hasRelative('Philip, 'Alice)
hasAuthorizedPersonID('Philip, '1001)
hasPatientID('Philip, 'P001)
hasBloodSugarLevelBeforeMeal('Philip, '256)
hasBodyTemperature('Philip, '104)
hasDiastolicBP('Philip, '88)
hasCaregiver('Philip, 'Maria)
logAlarm('Philip, 'KajangTown_Van3)
hasAmbulanceCallFor('Philip, 'KFCKajang)

```

Figure 6.6: Some of the ABox Property Axioms



```

Console
<terminated> PatientCareExample [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (May 19, 2016, 1:05:37 AM)
*****Horn Clause Rules*****

Tell14To8BurglarAlarm(?Beep) -> BurglarAlarm(?Beep)
Temperature(?temp),lessThan(?temp, "18" ) -> hasTemperature(?temp, "Cool")
OccupancySensor(?x) -> SmartHomerSensor(?x)
BloodSugarLevel(?bsl),Person(?p),hasBloodSugarLevelBeforeMeal(?p,?bsl), greaterThan(?bsl, "80" ),
lessThan(?bsl, "130" ) -> hasDBCategory(?p, "Controlled")
MotionDetector(?md),Person(?p) -> hasOccupancy(?p, "No")
Patient(?p),hasDBCategory(?p, "Hypoglycaemia") -> hasSituation(?p, "Emergency")
hasBloodPressure(?p, "Normal") -> Tell2To1hasBloodPressure(?p, "Normal")
Patient(?p),hasBloodPressure(?p, "Stage1hypertension"),hasDBCategory(?p, "EstablishedDiabetes"),
hasFever(?p, "High") -> hasSituation(?p, "OnCall")

```

Figure 6.7: Some of the Horn-clause Rules

Onto-HCR's main menu is shown in Figure 6.3 in which system prompts the user to input ontology. After executing several operations, the system produces TBox axioms from the ontology, some of them are shown in Figure 6.4. Figures 6.5 and 6.6 depict the extracted ABox axioms including classes and properties. Some of the OWL 2 RL and SWRL rules are shown in Horn-clause rule format in Figure 6.7. The smart environment ontology translated rules are given in appendix B. The main functions of Onto-HCR Translator are listed below:

- System prompts users to choose the ontology from the published source.
- Load the ontology file (in RDF/XML or OWL/XML format) as an input.
- Extract the set of logical axioms from the ontology, which can either be TBox axiom or ABox axiom.
- We use OWL parser to parse the ontology into OWL API objects which then extracts the set of TBox and ABox axioms.
- The set of TBox and ABox axioms are in the form of OWL 2 RL rules.
- We translate these set of axioms into a plain set of text in Horn-clause rules format.
- DL safe rule axioms are in the form of SWRL rules which are already in the form of Horn-clause rules.

- We also extract DL Safe rules (which are defined in Functional Syntax format) from ontology and then translate them into plain Horn-clause rule format.

These set of Horn-clause rules, translated from ontology using OWL API, are suitable to implement rule based reasoning strategy in context-aware multi-agent system.

6.5 A Smart Environment: Case Study

The main purpose of building this example system is to illustrate the use of the logical model \mathcal{L}_{DROCS} by focusing on automated analysis and formal verification using model checking techniques. We build the model for multi-agent non-monotonic context-aware system whose rules are derived from a smart environment domain ontology. We construct OWL 2 RL ontology domain of the given scenario using Protégé ontology editor to capture the static behavior of the the system while dynamic aspects of the system is depicted by SWRL rules. The example scenario is adopted from [Bikakis et al., 2010, Leijdekkers and Gay, 2006, Nabih et al., 2011], which is further extended based on the system users' requirements. This example system aims to facilitate residents in an intelligent home care environment that address residents' needs based on the current contexts. The aim is to create an automated assisted living environment for needy people to live a safe life and provide ease, comfort and security to them. This system provides the intelligent smart home environment where it is assumed that different sensing devices are installed to monitor the current situation of the person and the home. An agent, perhaps representing a particular device, could be implemented using simple rule-based technique.

In this system design, we consider a number of intelligent context-aware agents to monitor the current status of a person and the home environment. For example, a number of essential health care devices are considered to monitor a patient's vital information, which update status based on the current contexts. The bio-sensor agents are of the sort of bracelet(s) which is/are attached to the patient. These agents gets patient's current

senses such as blood pressure, blood sugar level, body temperature and pulse rate and notify Patient care (controller) agent after certain intervals of time. Whenever patient care agent receives most recently generated contexts from other agents, it immediately takes appropriate decision for the patient whether to declare an emergency situation or inform to the caregiver. The main inspiration is that each agent keeps the most recent values in the memory by overwriting conflicting context if it exists in the memory.

This smart home environment also considers some security agents to monitor unauthorized person's movement or prohibited activities at home. Figure 6.9 shows partial view of the ontology, and Figure 6.8 depicts smart space context-aware agents and their possible interactions. However, the complete set of rules are given in appendix A which are encoded using Maude LTL model checker in order to verify some interesting properties of this system.

The agents in Figure 6.8 are designed using the translated Horn-clause rules of the ontology. In this case study, we have considered 21 agents to model the smart home system by providing the basic health and safety needs for the residents. However, this case study can be extended by increasing the number of agents to provide additional health and safety features including self-indulgent facilities. Table 6.4 shows the list of agents and the number of rules for each agent and the second last column of the table shows agent's interaction to/from other agents.

6.5.1 Smart Environment Agents' Functions

The description of agent's tasks are given below:

1. **Patient Care Agent(PC):** is a centralized agent which receives patient's vital information from other agents after certain interval of time and decides about the current situation of the patient. If there is an emergency situation, it notifies emergency monitoring agent to take appropriate action. In case of a less emergency situation,

it informs to the caregiver to facilitate patients according to their need.

2. **Blood Pressure Agent (BP)**: Sends updated Blood Pressure value to the Patient Care agent after certain intervals of time. Blood Pressure range and types selected for the case study is standard which can be found online¹.
3. **Diabetes Tester (DB)**: Checks blood sugar level before meal, two hours after meal or at specified timings and sends current blood sugar value to the patient care agent.
4. **Body Temperature (BT)**: Sends updated body temperature values to the patient care agent.
5. **Pulse Rate Monitor (PM)**: Continuously monitors the pulse rate of the patient and sends updates to the patient care agent to take certain action in case of abnormalities.
6. **Ambulance Agent (NA)**: Immediately notifies ambulance staff to move to the GPS located point.
7. **Emergency Monitoring Agent (EM)**: This agent is informed by patient care agent in case of an emergency situation and at the same time this agent receives the GPS location of the patient and then notifies the Telephone agent to call an ambulance at GPS located coordinates point.
8. **OnCall Agent (OC)**: Upon receiving a request from the patient care agent and the GPS agent, this agent will automatically call the patient's caregiver or a nurse according to current situation of the patient.
9. **GPS sensor (GS)**: Mostly patients and elderly people stay at home. GPS sensor will be useful to detect the exact location of the person when he/she is outdoor and sends GPS coordinates for every movement to the OnCall, Emergency and Telephone agents when activated.

¹<http://www.mayoclinic.org/diseases-conditions/high-blood-pressure/in-depth/bloodpressure/art-20050982>

10. **Telephone Agent (TA)**: This agent may be activated upon receiving updates from the GPS sensor and Emergency monitoring agent and then automatically sends a call message with GPS coordinates to the ambulance agent.
11. **Motion Detector (MD)**: Has synchronization with the Image sensor agent and identifies the movement of authorized persons. It will raise burglar alarm in case of an unauthorized movement and notifies to the OnCall agent to call the caregiver of the patient.
12. **Temperature Level Sensor (TL)**: Checks the temperature of rooms, kitchen, corridor, etc and sends message to the Aircon sensor agent to increase or decrease temperature automatically.
13. **Gas Detector (GD)**: If this agent detects gas leakage, then it will activate burglar alarm automatically and will send immediate notification to the Telephone agent to call the Caregiver to take appropriate action.
14. **Glass Break Sensor (GS)**: Ensures the safety of all windows. It notifies the OnCall agent in case of glass breaking.
15. **Light Sensor (LS)**: This agent may be activated to turn on/off light based on person's occupancy. Light sensor switches on the light whenever it receives message of person's availability from occupancy sensor.
16. **Smoke Sensor (SS)**: It may alarm in case if smoke is detected. It detects the possibility of fire and fire status information is sent to OnCall agent to call caregiver.
17. **Aircon Sensor (AC)**: Detects the person's availability from the Occupancy sensor agent and turn on or off air conditioner. It may also increase or decrease temperature upon receiving request from the Temperature level sensor agent.
18. **Occupancy Sensor (OS)**: Monitors the presence of persons and sends messages to Light and Aircon sensor agents to act accordingly.

Seq.	Agent Name	Communicate To/From	No.of Rules
1	Patient Care Agent	2,3,4,5,7,8	47
2	Blood Pressure Agent	1	10
3	Diabetes Tester Agent	1	10
4	Body Temperature Agent	1	10
5	Pulse Monitor Agent	1	5
6	Ambulance Agent	10	2
7	Emergency Monitoring Agent	1,9,10 14,15,16,17	4
8	OnCall Agent	1,11,13,	9
9	GPS Sensor Agent	7,8,10,11	3
10	Telephone Agent	6,7,9	3
11	Caregiver Agent	8,9	2
12	Image Sensor Agent	13	4
13	Motion Detector Agent	8,12,19	6
14	Gas Detector Agent	8	3
15	Glass Break Sensor Agent	8	3
16	Smoke Sensor Agent	8	3
17	Relative Agent	8	2
18	Light Sensor Agent	19	4
19	Occupancy Sensor Agent	13,18,20	7
20	Aircon Sensor Agent	19,21	7
21	Temperature Level Sensor Agent	20	5

Table 6.4: Smart Environment Agent's Description

19. **Image Sensor (IS)**: Identifies visitor and checks whether he/she is authorized person or not and notify motion detector agent.
20. **Caregiver Agent (NA)**: Informs caregiver immediately upon receiving message from other agents.
21. **Relative Agent (NA)**: Notifies patient relative in case of any need as it receives message from other agents.

6.6 Encoding and verification of $\mathcal{L}_{\mathcal{DROCS}}$ model

In this section, we show how a $\mathcal{L}_{\mathcal{DROCS}}$ model (discussed in the previous chapter) can be encoded using Maude LTL model checker and how its interesting resource-bounded as well as conflicting properties can be verified automatically.


















Property assertions: 'Philip'	
Object property assertions	+
	hasBloodSugarLevelBeforeMeal '256
	hasRelative 'Alice
	hasPatientID 'P001
	hasAmbulanceCallFor 'KFCKajang
	hasBodyTemperature '104
	hasDiastolicBP '88
	hasSystolicBP '134
	logAlarm 'KajangTown_Van3
	hasCaregiver 'Maria
	hasPulse '120
Data property assertions	+
	isSmokeDetected "Yes"
	hasOccupancy "Yes"
	hasBloodPressure "Stage1Hypertension"
	hasSituation "OnCall"
	hasAirconFor "On"
	isGlassBroken "Yes"
Negative object property assertions	+
Negative data property assertions	+
	hasSituation "Emergency"

Figure 6.10: Individualized Smart Environment Ontology




















































Rules	
BodyTemperature(?temp, Person(?p), hasBodyTemperature(?p, ?temp), greaterThan(?temp, "95"), lessThan(?temp, "99") -> hasFever(?p, "Normal")	  
hasDBCategory(?p, "Hyperglycaemia") -> Tell 3, 1, hasDBCategory(?p, "Hyperglycaemia")	  
Temperature(?temp), greaterThan(?temp, "18"), lessThan(?temp, "25") -> hasTemperature(?temp, "Normal")	  
OccupancySensor(?x) -> SmartHomeSensor(?x)	  
AuthorizedPersonID(?apid), Person(?p), hasAuthorizedPersonID(?p, ?apid) -> isAuthorizedPerson(?p, "Yes")	  
Tell 7, 10, hasWarningSign(?p, ?loc) -> hasWarningSign(?p, ?loc)	  
Tell 8, 17, BurglarAlarm(?x) -> Alarm(?x)	  
Tell 2, 1, hasBloodPressure(?p, "Normal") -> hasBloodPressure(?p, "Normal")	  
Tell 10, 6, hasAmbulanceCallFor(?p, ?loc) -> hasAmbulanceCallFor(?p, ?loc)	  
hasBloodPressure(?p, "Prehypertension") -> Tell 2, 1, hasBloodPressure(?p, "Prehypertension")	  
Tell 9, 8, hasGPSLocation(?p, ?loc) -> hasGPSLocation(?p, ?loc)	  
Caregiver(?c), hasAlarmFor(?p, ?loc), hasCaregiver(?p, ?c) -> logAlarm(?c, ?p)	  
Tell 2, 1, hasBloodPressure(?p, "Hypotension") -> hasBloodPressure(?p, "Hypotension")	  
hasOccupancy(?p, "No") -> Tell 19, 18, hasOccupancy(?p, "No")	  
Tell 3, 1, hasDBCategory(?p, "Hypoglycaemia") -> hasDBCategory(?p, "Hypoglycaemia")	  
BodyTemperature(?temp), Person(?p), hasBodyTemperature(?p, ?temp), greaterThanOrEqual(?temp, "99"), lessThan(?temp, "101") -> hasFever(?p, "High")	  
DiastolicBP(?dbp), Person(?p), SystolicBP(?sbp), hasDiastolicBP(?p, ?dbp), hasSystolicBP(?p, ?sbp), greaterThan(?dbp, "90"), greaterThan(?sbp, "140"), lessThan(?dbp, "100"), lessThan(?sbp, "160") -> hasBloodPressure(?p, "StageI hypertension")	  

Figure 6.11: Some Rules of the Smart Environment Ontology

6.6.1 Maude Encoding

Maude has modular structuring mechanism and context-aware agents are encoded in a modular fashion. The Maude language essentially includes the set of sorts and subsorts with the operations on these sorts. As we have discussed the types of modules in Maude in Chapter 2, we have implemented the system using functional module and system module because both the set of system and functional modules are required to specify the system. We construct the generic functional module to define the set of sorts, subsorts, operations, variables, equations and strategies. In the functional module, agents are programmed using the set of rules which are represented by Maude equations. Each of these agents has its own local state (or configuration) and the structural formation of all local states (configurations) produce a global state (configurations) of the system.

In Maude, there is a number of library modules such as `NAT`, `BOOL` and `QID` etc. which have significant role in order to implement the system. For example; `NAT` and `BOOL` modules are useful for defining natural and Boolean values. `QID` module, on the other hand, is useful in defining the set of constant symbols (rule-based system's constant terms). These modules have been imported into the functional module. Maude variable is of the sort `QID` which is used to define variable symbols (rule-based system's variable terms). Both constants and variables are subsorts of sort `Term`. Considering the case study, a context is declared as an operator, for which its arguments are of the sort `Term` and returns the element of sort `Context`. Hence, `Context`'s arguments usually include constants and variables. All of these are of sort `Term`. Subsequently, the sort `context` is declared as a subsort of `WM` (also known as working memory). In Maude, Agents' rules are represented using Maude equations, one equation for each rule. The inference engine is implemented using a set of Maude rules.

6.6.2 Specifying and Verifying the System

We have considered three facets of the system while specifying and verifying its interesting properties using the Maude LTL model checker. This is partly because to observe and compare model checking performance and scalability.

The first system is modelled using five agents, namely 1, 2, 3, 4 and 5 which monitors the residents' (e.g., patient's) vital information such as Pulse Rate, Body Temperature, Blood Sugar Level etc. This system infers appropriate contexts based on the current contextual information of the patient whether e.g., there is an emergency situation or not, among others. In this system, the agents 2, 3, 4 and 5 are able to infer high-level contexts from sensed low-level contexts using Horn-clause rules in their knowledge-bases. These agents can classify current blood pressure, blood sugar, and pulse rate into different categories based on their current measurement values. For example, agent 2's knowledge-base contains rules including the following:

Person(?p), SystolicBP(?sbp), DiastolicBP(?dbp), hasSystolicBP(?p, ?sbp), hasDiastolicBP(?p, ?dbp), greaterThan(?sbp,'120), lessThan(?sbp,'140), greaterThan(?dbp,'80), lessThan(?dbp,'90) → hasBloodPressure(?p, 'Prehypertension) ;

Person(?p), SystolicBP(?sbp), DiastolicBP(?dbp), hasSystolicBP(?p, ?sbp), hasDiastolicBP(?p, ?dbp), greaterThan(?sbp,'140), lessThan(?sbp,'160), greaterThan(?dbp,'90), lessThan(?dbp,'100) → hasBloodPressure(?p, 'Stage1hypertension) ;

hasBloodPressure(?p, 'Stage1hypertension) → Tell(2,1, hasBloodPressure(?p, 'Stage1hypertension)) .

The first rule classifies that the person has blood pressure category *Prehypertension* if her Systolic Blood Pressure is greater than 120 and Diastolic Blood Pressure is greater than 80. That is, agent 2 may infer high-level context *hasBloodPressure('Philip, 'Prehypertension)* when the rule matches with the agent's working memory contexts, e.g., *Person('Philip), SystolicBP('134), DiastolicBP('88), hasSystolicBP('Philip, '134), hasDias-*

tolicBP('Philip, '88), greaterThan('134,'120), lessThan('134,'140), greaterThan('88,'80), lessThan('88,'90), and so on. The third rule is a communication rule of agent 2 through which it interacts with agent 1 and passes the context *hasBloodPressure('Philip, 'Prehypertension)* when it believes that *Philip* has *Prehypertension* at the moment. Similar to the above, agent 2 and all other agents in the system have other deduction and communication rules for other categories. It is important to note that the ontology driven rules do not have priority and a system designer is responsible to provide appropriate rule priorities while encoding the system into Maude.

In order to model the first scenario we have derived 105 Horn-clause rules from the smart environment ontology and distributed them to the agents as working memory facts and knowledge base rules. For example, the knowledge of agent 1 contains 45 rules, agent 2 is modelled using 10 rules, and so on. Whenever agent 1 receives most recently generated contexts from other agents, it infers current status of a patient and declares whether the patient has an emergency situation or not. The core inspiration is that each agent keeps the most recently derived contexts in the memory by overwriting an existing context, and this happens if agent's memory is full or a contradictory context arrives in the memory (even if the memory is not full). We verified a number of interesting resource-bounded properties of the system including the following non-conflicting contextual properties to see for example, when there is an emergency situation for a patient then the system should not produce non-emergency situation at the same time.

Prop1.1 : F(B₁ hasSituation('Philip,'Emergency))

Prop1.2 : F(B₁ Not(hasSituation('Philip,'Emergency)))

Prop1.3 : G(B₁ ~ (hasSituation('Philip,'Emergency) ∧ Not(hasSituation('Philip,'Emergency))))

The initial working memory facts (contexts) and rules are assigned to the agents in such a way that the system can infer both *hasSituation('Philip,'Emergency)* and *Not(hasSituation('Philip,'Emergency))* contexts that are conflicting. The operator B_i used in the properties to state that agent i believes a context (in other words certain context appears in

Properties	Number of state explored	Time required to verify properties (milliseconds)
<i>Prop1.1</i>	172	30ms
<i>Prop1.1</i>	280	45ms
<i>Prop1.2</i>	2332	384ms

Table 6.5: Experimental Results of the First System

the agent i 's working memory); and as usual G stands for always (globally), F stands for eventually (in the future), and X stands for next step. The truth of the first two properties ensure that indeed both these contexts can be inferred in the future, while the truth of the third property ensures that both of them never appear in the agent's memory at the same time. The above properties are verified as true and Maude reported the following results after verifying these properties as shown in Table 6.5. While verifying these properties minimum memory space required by agent 1 was 12 units and it exchanged 4 messages.

This second system that we consider for the verification is modelled using 11 agents, namely, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 and 11, and to model this second scenario 133 Horn-clause rules have been used. This system, in addition to inferring the residents' health status, interacts with various other agents to take appropriate actions. This enhances the services and making system more complex. For example, agent 1 interacts with Emergency monitoring agent and OnCall agent which in turn interacts with various other agents to locate GPS coordinate points to call Ambulance via Telephone agents. Upon receiving message from agent 10, ambulance could move to GPS located point to rescue patient. In addition, Caregiver is also notified by OnCall agent about the emergency situation with GPS coordinates point of the patient. We verified a number of interesting resource-bounded properties of this system including those we considered above in the first system.

Prop2.1 : $F(B_1 \text{ hasSituation}('Philip, 'Emergency))$

Prop2.2 : $F(B_1 \text{ Not}(\text{hasSituation}('Philip, 'Emergency)))$

Prop2.3 : $G(B_1 \sim (\text{hasSituation}('Philip, 'Emergency) \wedge \text{Not}(\text{hasSituation}('Philip, 'Emergency))))$

Prop2.4 : $G(B_8(\text{Tell}(1, 8, \text{hasSituation}('Philip, 'Oncall))) \wedge$

Properties	Number of state explored	Time required to verify properties (milliseconds)
<i>Prop2.1</i>	282	60ms
<i>Prop2.2</i>	640	131ms
<i>Prop2.3</i>	4336	1084ms
<i>Prop2.4</i>	4336	1050ms

Table 6.6: Experimental results of the Second System

$$(Tell(9, 8, hasGPSLocation('Philip', KFCKajangTown)) \rightarrow X^n B_8 Tell(8, 11, hasAlarmFor('Philip', KFCKajangTown)))$$

The fourth property above specifies that whenever agents 1 and 9 tell agent 8 that the *Philip* has *OnCall* situation and his GPS location is at *KFCKajangTown*, within n time steps agent 8 sending an alarming message to agent 11. All the above properties are verified as true and Maude reported the following results after verifying these properties as shown in Table 6.6. While verifying these properties minimum memory space required by agent 1 was 14 and agent 8 by 8 units and the value of n was 4 (i.e., within 4 time steps agent 8 sending an alarming message to agent 11). The messages that the agents exchanged were: agent 1: 5, agent 2: 1, agent 3: 1, agent 4: 1, agent 5: 1, agent 6: 1, agent 7: 3, agent 8: 3, agent 9: 2, agent 10: 2, and agent 11: 1.

The third system that we consider for the verification is modelled using all the 21 agents, and to model this scenario 201 Horn-clause rules have been used. This system models very complex scenarios and deals with a very high level of combinatorial aspects. It includes some smart home sensor agents to provide ease, comfort, security and healthy life in the smart home. In this system, the sensor agents (agents 12 – 21) monitor the basic safety measures at home and inform to relatives of the patient for any kind of mishap occurrence in the smart home. For example, burglar alarm will ring in case e.g., smoke is detected by the Smoke sensor agent, and then OnCall agent immediately interact with the Relative agent to take appropriate actions. This system also checks the existence of a person in a room and automatically switch on/off the light and air-condition based on the current contexts. So saving energy is the additional requirement of the system. However, by

Properties	Number of state explored	Time required to verify properties (milliseconds)
<i>Prop3.1</i>	379210	165461ms
<i>Prop3.2</i>	379210	164321ms

Table 6.7: Experimental results of the Third System

adding more agents the system designer can make the system much more complex. We verified a number of interesting resource-bounded properties of this system including the following:

$$\begin{aligned}
\text{Prop3.1 : } & G(B_8(\text{Tell}(1, 8, \text{hasSituation}('Philip, 'Oncall))) \wedge \\
& (\text{Tell}(9, 8, \text{hasGPSLocation}('Philip, 'KFCKajangTown)) \rightarrow \\
& X^n B_8 \text{Tell}(8, 11, \text{hasAlarmFor}('Philip, 'KFCKajangTown))) \\
\text{Prop3.2 : } & G(B_{11}(\text{Tell}(8, 11, \text{hasAlarmFor}('Philip, 'KFCKajangTown))) \\
& \rightarrow X^n B_{11} \text{logAlarm}('Alice, 'Philip))
\end{aligned}$$

The first property is same as *Prop2.4* above, while second property above specifies that whenever agent 8 tells agent 11 that *Philip* has alarming situation and his GPS location is at *KFCKajangTown*, within n time steps *Alice* (caregiver agent 11) noticing this. Both the above properties are verified as true and Maude reported the following results after verifying these properties as shown in Table 6.7. The value of n in *Prop3.2* is 2. However, when we assign a value to n which is less than 4 in *Prop2.4*, and less than 2 in *Prop3.2* the properties are verified as false and the model checker returns counterexamples. Similarly, when we assign a value to memory size (or message counter) which is less than the minimal required value, properties are verified as false. This also ensures the correctness of the encoding in that model checker does not return true for arbitrary values of n , memory and message counters. Note that, verification of true formulas take longer than verification of false formulas since a model checker will find a counterexample faster than it takes to explore the whole model.

6.7 Conclusion

In summary, we have described how ontologies can be translated into Horn-clause rules. To directly translate ontological knowledge into Horn-clause rules, we have listed the main functions of the *Onto-HCR* tool developed for the translation process. We also have described a comprehensive case study of a smart environment to model context-aware resource-bounded non-monotonic reasoning based system and verify its resource-bounded as well as non-monotonic properties using the Maude LTL model checker. The scalability and expressiveness is evaluated using the above mentioned case study by considering three facets of the system. In the next chapter, we provide a concise summary of the thesis and then discuss some possible directions for future work.

Chapter 7

Conclusion and Future Work

This thesis aims to develop logical frameworks for the representation and reasoning about resource-bounded context-aware systems, which allow us to investigate, for example, whether context-aware agents can infer certain contexts or they never infer conflicting contexts while they are resource-bounded. This chapter is divided into two sections. We briefly summarize the core contribution of this thesis in the first section, and in the second section we provide a brief summary of key topics for some future works.

7.1 Summary

This research has introduced a new vision of context-aware systems considering today's modern world complex problem solving in a highly decentralized environment. As context-aware systems are adaptive in nature and mostly run on smart devices. However, many challenges might arise when these devices exchange information among themselves in order to solve a problem with their limited computational and communication resources. In this thesis, we have presented systematic formal logical frameworks for modelling and verifying resource-bounded context-aware rule-based multi-agent systems. Where agents reason using ontology-driven first order Horn clause rules. We have realized the significance and needs of these formalisms based on the literature review. As this research work

has incorporated many disciplines from the literature, so it is vital to have a brief survey on each of them.

The literature review presented in Chapters 2, 3 has focused on reasoning formalisms for the semantic web. We have mainly focused on ontology and SWRL because the work presented in this thesis is based on ontology-driven rule based reasoning agents where rules are derived from OWL 2 RL and SWRL. We have reviewed literature on description logics, web ontology languages (OWL) and SWRL. Description logic has been considered as one of the most expressive formal languages having capability to perform reasoning about knowledge in an application domain. Description logic is based on ontology languages that are often used for context representation and reasoning. However, this logic may not be applicable for non-monotonic rule-based reasoning systems. For non-monotonic reasoning, we have considered defeasible reasoning owing to its efficient reasoning capability, low computational complexity, and its focus on implementability. Defeasible reasoning is used to reason inconsistent and incomplete information. We have surveyed literature on monotonic as well as non-monotonic reasoning based logical formalisms including rule-based reasoning and the semantic web technologies. In doing so, we realized their efficacy towards context-aware systems. The ultimate purpose of considering this literature is to craft a comparative study that showcases the relevant and significant information regarding context-aware logical frameworks.

In Chapter 4, we have presented a formal logical framework for modelling and verifying context-aware multi-agent systems where agents reason using ontology-driven first order Horn-clause rules. In this work, we considered space requirement for reasoning in addition to the time and communication resources. We extended CTL* with belief and communication modalities, and the resulting logic \mathcal{L}_{OCS} allows us to describe a set of rule-based reasoning agents with bound on time, memory and communication. We modelled an ontology-based context-aware system and verified its resource-bounded properties. There is one drawback of this logic is that it is based on monotonic reasoning where

beliefs of an agent cannot be revised based on some contradictory evidence.

In Chapter 5, we have proposed a logical framework for modelling context-aware systems as multi-agent non-monotonic rule-based agents, and the resulting logic $\mathcal{L}_{\mathcal{DROCS}}$ allows us to describe a set of ontology-driven rule-based non-monotonic reasoning agents with bounds on time, memory, and communication. Agents use defeasible reasoning technique to reason with inconsistent information. The proposed framework allows us to determine how much time (measured as rule-firing cycles) are required to generate certain contexts, how many messages must be exchanged among agents, and how much space (memory) is required for an agent for the reasoning.

In Chapter 6, we have provided a suitable translation technique for translating ontological knowledge into Horn-clause rules. We have also developed a tool, *Onto-HCR*, which translates semantic knowledge into Horn-clause rules format. we have modelled and developed a case study of a smart environment to model context-aware resource-bounded non-monotonic reasoning system and verified its resource-bounded as well as non-monotonic properties using the Maude LTL model checker. The scalability and expressiveness is evaluated by considering three facets of the system.

7.2 Future Work

The research work presented in this thesis can be extended in a number of ways which could be addressed as future work.

7.2.1 Extending Logic $\mathcal{L}_{\mathcal{DROCS}}$ Using Multi-Context System (MCS)

One direction is to extend our work $\mathcal{L}_{\mathcal{DROCS}}$ with the incorporation of multi-context system that will be used to model and state interesting properties of the distributed systems to be verified in a highly decentralized environment. Most works on multi-context systems stem from non-monotonic reasoning in Ambient Intelligence [Bikakis et al., 2008, Brewka

et al., 2007, Benslimane et al., 2006]. To our knowledge, there have been no formal logical frameworks which incorporate multi-context system with resource-bounded context-aware multi-agent system. For this, we already have done some initial work. In this section we briefly describe how we can extend the $\mathcal{L}_{\mathcal{DROCS}}$ with the incorporation of multi-context systems. A multi-context system includes of a set of contexts and a set of inference rules that allows information to flow among different contexts. In MCS, each context is defined as a self-contained knowledge source which includes the set of axioms and inference rules to model the system and perform reasoning. It is a very powerful framework to integrate various distributed knowledge sources and to model the flow of information among themselves.

Literature highlighted many definitions of multi-context systems [Eiter et al., 2014, Ghidini and Giunchiglia, 2001, Brewka et al., 2007]. In [Brewka et al., 2007], Brewka et al. define multi-context system as a number of people, agents and databases to describe the available information from a set of contexts and inference rules and specify the information flow among these contexts. In [Benslimane et al., 2006], Benslimane et al. have described ontology as a context, which is itself an independent self-contained knowledge source having a set of axioms and inference rules with its own reasoner to perform reasoning. We consider the later definition because context-aware agents need to acquire information from different semantic knowledge sources which are interlinked using interlinking axioms (bridge rules) in order to achieve the desired goals.

In the proposed logical framework, non-monotonic context-aware agents will acquire contexts either from a single ontology or multiple ontologies based on the design of the system. Ontological knowledge such as OWL 2 RL, SWRL and bridge rules will be written based on the available information (acquired by the sensors/agents) stored in the ontology. These rules will be static and set by the system designer at the design time of the system. In this system, each agent will have a simple program to perform a specific task and each agent in the system might acquire a set of specified contexts from one ontology

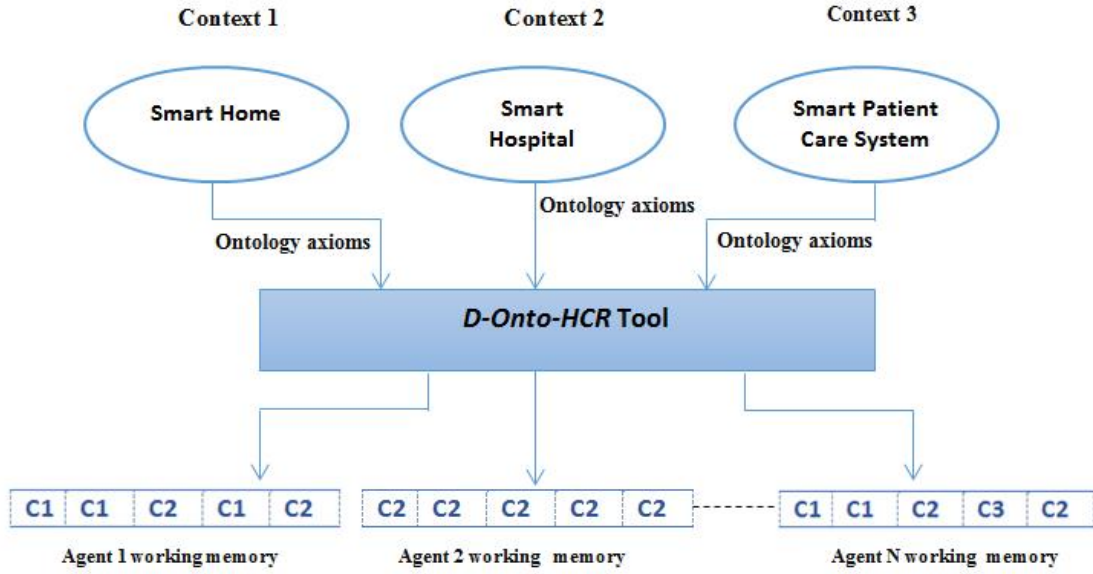


Figure 7.1: Multi-context awareness in the working memory of agent i

or multiple ontologies to derive target information.

This system would enable context-aware agents to acquire contexts from distributed knowledge sources and then perform reasoning using a set of strict, defeasible and bridge rules. This system will be modelled as non-monotonic context-aware multi-agent system. We provide conceptual mapping of the proposed model in Figure 7.1, which is illustrated as follows. Agent 1's working memory contains the contexts $C1$ and $C2$ which are instances of smart home and smart hospital. The working memory of agent 2 has contexts only from smart hospital ontology whereas the working memory of agent N includes the instances of all contexts in the system. Each agent in the system will be represented by a triple $(\mathcal{R}, \mathcal{F}, \succ)$, where \mathcal{F} is a finite set of facts contained in the working memory, $\mathcal{R} = (\mathcal{R}^s, \mathcal{R}^d, \mathcal{R}^{br})$ is a finite set of strict, defeasible rules and bridge rules, and \succ is a superiority relation on \mathcal{R} . Strict rules (\mathcal{R}^s) are non-contradictory whereas defeasible rules (\mathcal{R}^d) can be defeated based on contrary evidence. Bridge rules (\mathcal{R}^{br}) are non-contradictory rules which represent the distributed knowledge base concepts. These rules are fired based on their predefined priorities which is set by the system designer. This system will con-

tinue to derive contextual information until the desired goal is achieved.

7.2.2 Contextualizing Ontologies

In this section, we aim to provide a concrete methodology of contextualizing ontologies for the proposed logical framework discussed in the previous section. To model the system for context-aware non-monotonic reasoning agents, we extract heterogeneous contextual information from multiple ontologies with the intention of preserving the identity and independence of each specialized domain ontology. Distributed description logic is a very suitable modelling approach which syntactically and semantically inter-connect different domain ontologies through semantic mappings and express the relationships among them [Borgida and Serafini, 2003]. DDL is a set of DL knowledge bases in which each DL Knowledge base axioms (TBox and ABox) is mapped from its corresponding ontology.

To model distributed domains for the proposed system, we develop three ontologies named as Smart Patient Care System (O_{SPC}), Smart Home (O_{SHO}) and Smart Hospital (O_{SHP}) which have their corresponding DL knowledge bases as \mathcal{DL}_{SPC} , \mathcal{DL}_{SHO} and \mathcal{DL}_{SHP} respectively. We already have discussed DL ontology mapping in our previous work [Rakib and Haque, 2014]. Additionally, we construct the bridge rules (or inter-ontology axioms) which are semantically mapped using distributed DL Knowledge base. Figure 7.2 depicts the extracts of class hierarchies of three ontologies. Some of the bridge rules are given as:

$$O_{SPC} : Patient \sqsubseteq_{\rightarrow} O_{SHO} : AuthorizedPerson. \quad (1)$$

$$O_{SPC} : Nurse \sqsubseteq_{\rightarrow} O_{SHO} : AuthorizedPerson. \quad (2)$$

$$O_{SPC} : Nurse \sqsubseteq_{\rightarrow} O_{SHP} : ParamedicalStaff. \quad (3)$$

$$O_{SPC} : CallAmbulance \sqsubseteq_{\rightarrow} O_{SHP} : AmbulatoryClinic. \quad (4)$$

Bridge rules 1 and 2 show the relationship between O_{SPC} and O_{SHO} and rules 3 and 4 show the relationship between O_{SPC} and O_{SHP} . Rule 1 states that the patient from patient care ontology is an authorized persons in the smart home. Rule 2 and 3 express that nurse

from patient care ontology is authorized person in smart home and at the same time nurse is a paramedical staff in the smart Hospital. These rules can also be represented in first order form, for example; the first rule is re-written as

$$Patient(?p) \mapsto AuthorizedPerson(?p) \quad (1)$$

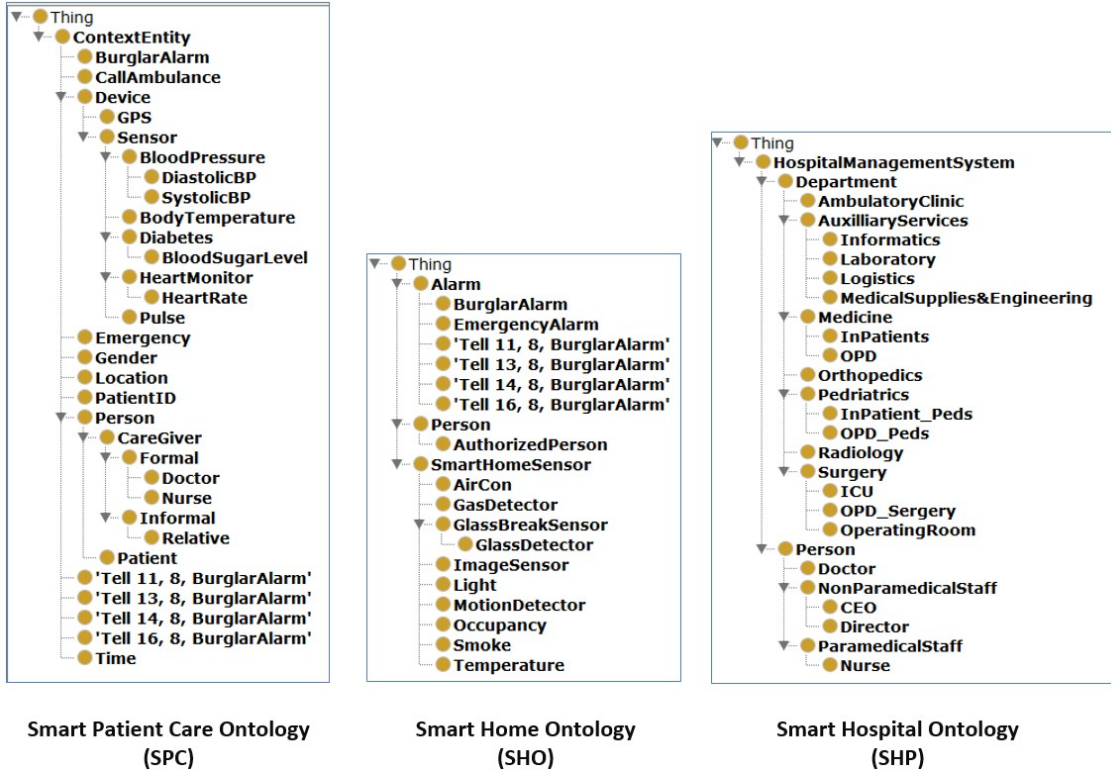


Figure 7.2: Class hierarchy of Smart Environment Ontologies

We model the system using OWL 2 RL ontologies (including bridge and SWRL rules) and extract the set of rules from different ontologies to design non-monotonic context-aware rule-based agents.

7.2.3 Distributed Semantic Knowledge Translator (*D-Onto-HCR*)

To extract the rules from different ontologies, we develop an initial version of OWL-API based translator, *D-Onto-HCR*, which takes ontologies as input and then translates the set of axioms (in OWL 2 RL and SWRL form) into a plain text of Horn-clause rules.

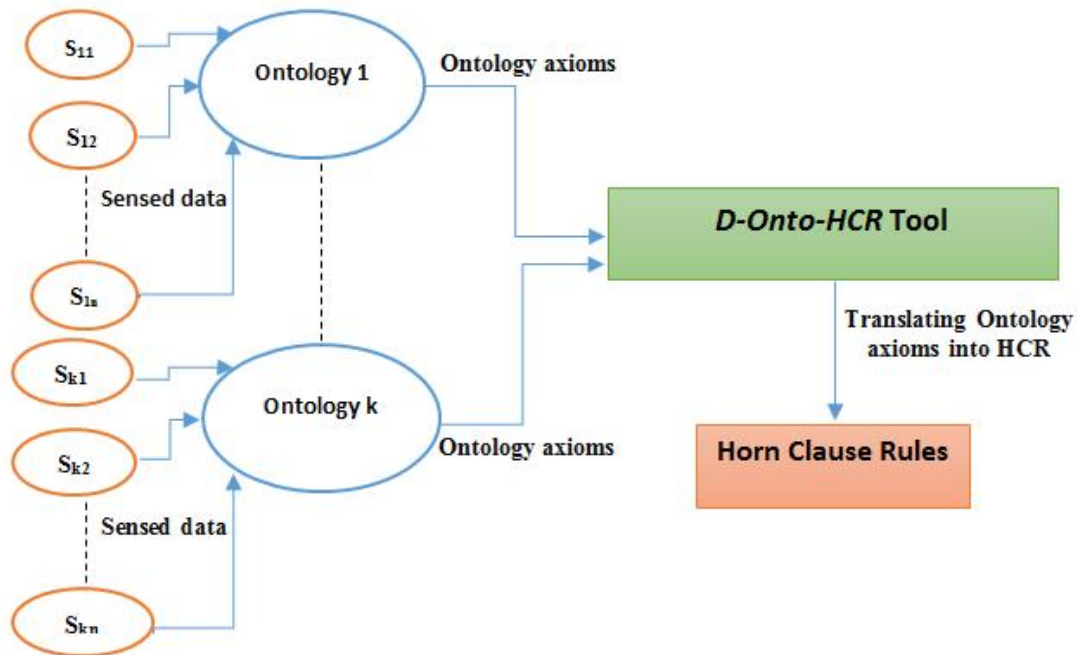


Figure 7.3: Distributed Semantic Knowledge Translation Process

The design of the OWL API corresponds to the OWL 2 Structural Specification and this dynamic design model allows developers to provide flexible implementations for major components of the system. In OWL API, the names and hierarchies for the axioms, class expressions and entities correspond to the OWL structural specification. Indeed, there is a proximal one to one translation between OWL API model interfaces and the OWL 2 Structural Specification, implying that this becomes easier to correlate the high level OWL 2 specification with the design of the OWL API [Horridge and Bechhofer, 2009]. To extract ontology axioms and facts, we use OWL-API [Horridge and Bechhofer, 2009] to parse the ontology. Protégé [Protégé, 2011] ontology editor allows SWRL rules to be written in Horn-clause rule format but practically these rules are written in functional syntax which are in DL-Safe rule form. *D-Onto-HCR* translates DL-safe rules axioms into Horn-clause rules format. Additionally, this translator extracts concepts and roles from different ontologies and maps them correspondingly in the form of bridge rules which are transformed in OWL 2 RL rule format. These rules are then translated into a set

of plain Horn-clause rules format. Figure 7.3 shows the distributed semantic knowledge translation process and the output generated from *D-Onto-HCR* is given in Figure 7.4.

```

Console
<terminated> App (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (May 24, 2016, 11:20:43 PM)

Welcome to D-Onto-HCR
=====
List of ontologies
=====
1. Smart Patient Care Ontology
2. Smart Home Ontology
3. Smart Hospital Ontology

Now Loading Ontologies...

Console
<terminated> App (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (May 24, 2016, 11:20:43 PM)

Now Loading Smart Hospital Ontology
=====

*****TBox axioms*****

Nurse -> ParamedicalStaff
NonParamedicalStaff -> Person
AmbulatoryClinic -> Department
Informatics -> AuxilliaryServices
Surgery -> Department
Logistics -> AuxilliaryServices
Medicine -> Department
Person -> HospitalManagementSystem

Console
<terminated> App (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (May 24, 2016, 11:20:43 PM)

Bridge Rules
=====
Patient(?p) -> AuthorizedPerson(?p)
Nurse(?n) -> ParamedicalStaff(?n)
Nurse(?n) -> AuthorizedPerson(?n)
Relative(?r) -> AuthorizedPerson(?r)
CallAmbulance(?amb) -> AmbulatoryClinic(?amb)
Tell11To8BurglarAlarm(?alarm) -> BurglarAlarm(?alarm)
Tell13To8BurglarAlarm(?alarm) -> BurglarAlarm(?alarm)

```

Figure 7.4: *D-Onto-HCR* Output

The new tool *D-Onto-HCR* should have the following features:

- System should prompt users to choose the ontologies from the published source.
- Load the ontology files (in RDF/XML or OWL/XML format) as an input.
- Extract the set of logical axioms from the ontology, which can either be TBox axiom or ABox axiom.
- We use OWL parser to parse the ontology into OWL API objects which then extracts the set of TBox and ABox axioms.
- The set of TBox and ABox axioms are in the form of OWL 2 RL rules.

- We translate these set of axioms into a set of plain text in Horn-clause rules format.
- DL safe rule axioms are in the form of SWRL rules which are already in the form of Horn-clause rules.
- We also extract DL Safe rules (which are defined in Functional Syntax format) from ontology and then translate them into plain Horn-clause rule format.
- The inter-ontology axioms are extracted from different ontologies and are transformed as bridge rules.

These set of Horn-clause rules, translated from ontologies using OWL API, are suitable to implement rule based context-aware multi-agent systems.

7.2.4 Potential Application Framework using Context-aware Resource-bounded Devices

The frameworks presented in this thesis are not practically implemented yet, so another possible direction is to implement these frameworks using Android platform. Application systems can be developed using Android platform in which smartphones act as context-aware agents. The potential application would be autonomous in a sense that the system would act independently on behalf of the user. This application would be installed and run on smartphones. It would acquire contextual information automatically from its specified domain, perform reasoning in order to derive the goal and then adapt its behavior accordingly. This system would be very useful for safety critical domains such as disaster recovery, emergency situations, elder care systems, traffic control system, among others.

7.3 Conclusion

This chapter has two core sections. In the first section, we have recapitulated the core contribution of the thesis and has documented the relevant literature to figure out the

significant gaps to be filled with this research work. In the second section, we discussed briefly the future work which is intended to be undertaken as an extended work in a number of ways. One possible direction is to extend the logical framework $\mathcal{L}_{\mathcal{DROCS}}$ with the incorporation of multi-context systems. Another possible direction discussed, in this chapter, is to model the system using distributed description logics considering multiple ontologies. At the end of this chapter, we discussed the potential application framework to be developed using context-aware resource-bounded devices.

Bibliography

- [Abowd et al., 1999] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. (1999). Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer.
- [Agre, 2001] Agre, P. E. (2001). Changing places: Contexts of awareness in computing. *Hum.-Comput. Interact.*, 16(2):177–192.
- [Albore et al., 2006] Albore, A., Alechina, N., Bertoli, P., Ghidini, C., Logan, B., and Serafini, L. (2006). Model-checking memory requirements of resource-bounded reasoners. In *AAAI*, volume 6, pages 213–218.
- [Alechina et al., 2007] Alechina, N., Bertoli, P., Ghidini, C., Jago, M., Logan, B., and Serafini, L. (2007). *Model Checking and Artificial Intelligence: 4th Workshop, MoChArt IV, Riva del Garda, Italy, August 29, 2006, Revised Selected and Invited Papers*, chapter Verifying Space and Time Requirements for Resource-Bounded Agents, pages 19–34. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Alechina et al., 2006] Alechina, N., Jago, M., and Logan, B. (2006). Modal logics for communicating rule-based agents. In *ECAI*, volume 6, pages 322–326.
- [Alechina et al., 2008] Alechina, N., Jago, M., and Logan, B. (2008). Preference-based belief revision for rule-based agents. *Synthese*, 165(2):159–177.
- [Alechina et al., 2009a] Alechina, N., Logan, B., Nga, N., and Rakib, A. (2009a). Verifying time and communication costs of rule-based reasoners. In Peled, D. and

- Wooldridge, M., editors, *Model Checking and Artificial Intelligence*, volume 5348 of *Lecture Notes in Computer Science*, pages 1–14. Springer Berlin Heidelberg.
- [Alechina et al., 2009b] Alechina, N., Logan, B., Nguyen, H. N., and Rakib, A. (2009b). Reasoning about other agents beliefs under bounded resources. In *Knowledge Representation for Agents and Multi-Agent Systems*, pages 1–15. Springer.
- [Alechina et al., 2009c] Alechina, N., Logan, B., Nguyen, H. N., and Rakib, A. (2009c). Verifying time, memory and communication bounds in systems of reasoning agents. *Synthese*, 169(2):385–403.
- [Alpern and Schneider, 1985] Alpern, B. and Schneider, F. B. (1985). Defining liveness. *Information Processing Letters*, 21(4):181 – 185.
- [Alur et al., 1998] Alur, R., Henzinger, T. A., Mang, F. Y. C., Qadeer, S., Rajamani, S. K., and Tasiran, S. (1998). *Computer Aided Verification: 10th International Conference, CAV'98 Vancouver, BC, Canada, June 28 – July 2, 1998 Proceedings*, chapter MOCHA: Modularity in model checking, pages 521–525. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Antoniou, 2002] Antoniou, G. (2002). A nonmonotonic rule system using ontologies. In *RuleML*. Proceedings of the First International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML 2002), volume 60 of CEUR Workshop Proceedings.
- [Antoniou and Bikakis, 2007] Antoniou, G. and Bikakis, A. (2007). Dr-prolog: A system for defeasible reasoning with rules and ontologies on the semantic web. *Knowledge and Data Engineering, IEEE Transactions on*, 19(2):233–245.
- [Antoniou et al., 1999a] Antoniou, G., Billington, D., Governatori, G., and Maher, M. J. (1999a). On the modeling and analysis of regulations. In *Australian Conference Information Systems*, pages 20–29.

- [Antoniou et al., 2001] Antoniou, G., Billington, D., Governatori, G., and Maher, M. J. (2001). Representation results for defeasible logic. *ACM Transactions on Computational Logic*, 2(2):255–287.
- [Antoniou et al., 1999b] Antoniou, G., Maher, M. J., Billington, B., and Governatori, G. (1999b). A comparison of sceptical naf-free logic programming approaches. In *Logic Programming and Nonmonotonic Reasoning*, pages 347–356. Springer.
- [Antoniou et al., 2005] Antoniou, G., Skylogiannis, T., Bikakis, A., and Bassiliades, N. (2005). Dr-brokering-a defeasible logic-based system for semantic brokering. In *e-Technology, e-Commerce and e-Service, 2005. IEEE'05. Proceedings. The 2005 IEEE International Conference on*, pages 414–417. IEEE.
- [Artale, a] Artale, A. Formal methods: Linear temporal logic. <http://www.inf.unibz.it/~artale/>. Accessed: 2016-07-04.
- [Artale, b] Artale, A. Ltl: Linear-time logic. http://en.wikipedia.org/wiki/Linear_temporal_logic. Accessed: 2016-05-04.
- [Augusto and Simari, 2001] Augusto, J. C. and Simari, G. R. (2001). Temporal defeasible reasoning. *Knowledge and information systems*, 3(3):287–318.
- [Baader, 2003] Baader, F. (2003). *The description logic handbook: theory, implementation, and applications*. Cambridge university press.
- [Baader et al., 2003] Baader, F., Küsters, R., and Wolter, F. (2003). Extensions to description logics. In *The description logic handbook*, pages 219–261. Cambridge University Press.
- [Baader and Nutt, 2003] Baader, F. and Nutt, W. (2003). Basic description logics. In *Description logic handbook*, pages 43–95.
- [Baier et al., 2008] Baier, C., Katoen, J.-P., et al. (2008). *Principles of model checking*, volume 26202649. MIT press Cambridge.

- [Baldauf et al., 2007] Baldauf, M., Dustdar, S., and Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277.
- [Baral and Gelfond, 1994] Baral, C. and Gelfond, M. (1994). Logic programming and knowledge representation. *The Journal of Logic Programming*, 19:73–148.
- [Barnat et al., 2006] Barnat, J., Brim, L., Černá, I., Moravec, P., Ročkai, P., and Šimeček, P. (2006). *Computer Aided Verification: 18th International Conference, CAV 2006, Seattle, WA, USA, August 17-20, 2006. Proceedings*, chapter DiVinE – A Tool for Distributed Verification, pages 278–281. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Bassiliades et al., 2004] Bassiliades, N., Antoniou, G., and Vlahavas, I. (2004). A defeasible logic reasoner for the semantic web. In *Rules and Rule Markup Languages for the Semantic Web*, pages 49–64. Springer.
- [Bassiliades and Vlahavas, 2003] Bassiliades, N. and Vlahavas, I. P. (2003). Capturing rdf descriptive semantics in an object oriented knowledge base system. In *Proc. International Word Wide Web Conference*.
- [Bechhofer, 2007] Bechhofer, S. (2007). Programming to the owl api: Introduction, university of manchester. <http://owlapi.sourceforge.net/SKB-SemTech-OWLAPI-6up.pdf>. Accessed: 2015-12-15.
- [Bechhofer et al., 2003] Bechhofer, S., Volz, R., and Lord, P. (2003). Cooking the semantic web with the owl api. In *The Semantic Web-ISWC 2003*, pages 659–675. Springer.
- [Benslimane et al., 2006] Benslimane, D., Arara, A., Falquet, G., Maamar, Z., Thiran, P., and Gargouri, F. (2006). *Advances in Information Systems: 4th International Conference, ADVIS 2006, Izmir, Turkey, October 18-20, 2006. Proceedings*, chapter Contextual Ontologies, pages 168–176. Springer Berlin Heidelberg, Berlin, Heidelberg.

- [Benthem, 2010] Benthem, J. V. (2010). Modal logic for open minds. *CSLI lecture notes*, Center for the Study of Language and Information.
- [Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., Lassila, O., et al. (2001). The semantic web. pages 34–43.
- [Bettini et al., 2010] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., and Riboni, D. (2010). A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2):161 – 180. Context Modelling, Reasoning and Management.
- [Bikakis and Antoniou, 2010] Bikakis, A. and Antoniou, G. (2010). Defeasible contextual reasoning with arguments in ambient intelligence. *Knowledge and Data Engineering, IEEE Transactions on*, 22(11):1492–1506.
- [Bikakis et al., 2010] Bikakis, A., Antoniou, G., and Hasapis, P. (2010). Strategies for contextual reasoning with conflicts in ambient intelligence. *Knowledge and Information Systems*, 27(1):45–84.
- [Bikakis et al., 2008] Bikakis, A., Antoniou, G., and Hassapis, P. (September 19, 2008). Distributed defeasible reasoning in ambient intelligence. *Preprint submitted to Data and Knowledge Engineering*.
- [Blackburn et al., 2002] Blackburn, P., De Rijke, M., and Venema, Y. (2002). *Modal Logic: Graph. Darst*, volume 53. Cambridge University Press.
- [Bordini et al., 2003] Bordini, R. H., Fisher, M., Pardavila, C., Visser, W., and Wooldridge, M. (2003). *Computer Aided Verification: 15th International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003. Proceedings*, chapter Model Checking Multi-Agent Programs with CASP, pages 110–113. Springer Berlin Heidelberg, Berlin, Heidelberg.

- [Borgida and Serafini, 2003] Borgida, A. and Serafini, L. (2003). Distributed description logics: Assimilating information from peer sources. In *Journal on Data Semantics I*, pages 153–184. Springer.
- [Botoeva, 2014] Botoeva, E. (2014). *Description logic knowledge base exchange*. PhD thesis, Ph. D. thesis, Free University of Bozen-Bolzano.
- [Brewka et al., 2007] Brewka, G., Roelofsen, F., and Serafini, L. (2007). Contextual default reasoning. In *IJCAI*, pages 268–273.
- [Brown, 1996] Brown, M. (1996). Supporting user mobility. In *Mobile Communications*, IFIP The International Federation for Information Processing, pages 69–77. Springer US.
- [Brown et al., 1997] Brown, P., Bovey, J., and Chen, X. (1997). Context-aware applications: from the laboratory to the marketplace. *Personal Communications, IEEE*, 4(5):58–64.
- [Calvanese and De Giacomo, 2003] Calvanese, D. and De Giacomo, G. (2003). Expressive description logics. In *The Description Logic Handbook*, pages 178–218. Cambridge University Press.
- [Castro and Muntz, 1999] Castro, P. and Muntz, R. (1999). Using context to assist in multimedia object retrieval. In *First International Workshop on Multimedia Intelligent Storage and Retrieval Management*. Citeseer.
- [Chen et al., 2003] Chen, H., FININ, T., and JOSHI, A. (2003). An ontology for context-aware pervasive computing environments. *The Knowledge Engineering Review*, 18:197–207.
- [Chen and Tolia, 2001] Chen, H. and Tolia, S. (2001). Steps towards creating a context-aware software agent system. *Technical report, Hewlett Packard Labs, Palo Alto HPL-2001*.

- [Clavel et al., 2007] Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., and Talcott, C. (2007). All about maude, a high-performance logical framework, volume 4350 of *lecture notes in computer science*.
- [connor et al, 2005] connor et al, O. (2005). Writing rules for the semantic web using swrl and jess. *Protégé With Rules Workshop 2005, Madrid, Spain*.
- [Cooperstock et al., 1995] Cooperstock, J. R., Tanikoshi, K., Beirne, G., Narine, T., and Buxton, W. A. S. (1995). Evolution of a reactive environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 170–177, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.
- [Creignou et al., 2012] Creignou, N., Meier, A., Vollmer, H., and Thomas, M. (2012). The complexity of reasoning for fragments of autoepistemic logic. *ACM Transactions on Computational Logic (TOCL)*, 13(2):17.
- [Daconta et al., 2004] Daconta, M. C., Smith, K. T., and Oerst, L. J. (2004). The semantic web: a guide to the future of xml, web services, and knowledge management. *Computing Reviews*, 45(12):778–779.
- [Daniele et al., 2007] Daniele, L., Costa, P. D., and Pires, L. F. (2007). Towards a rule-based approach for context-aware applications. In *EUNICE*, volume 4606 of *Lecture Notes in Computer Science*, pages 33–43. Springer.
- [Dastani et al., 2005] Dastani, M., Governatori, G., Rotolo, A., and Van Der Torre, L. (2005). Programming cognitive agents in defeasible logic. In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 621–636. Springer.
- [Dey and Abowd, 2000] Dey, A. and Abowd, G. (2000). Cybrereminder: A context-aware system for supporting reminders. In Thomas, P. and Gellersen, H.-W., editors, *Handheld and Ubiquitous Computing*, volume 1927 of *Lecture Notes in Computer Science*, pages 172–186. Springer Berlin Heidelberg.

- [Dey et al., 2001] Dey, A. K., Abowd, G. D., and Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-computer interaction*, 16(2):97–166.
- [Dey et al., 1998] Dey, A. K., Abowd, G. D., and Wood, A. (1998). Cyberdesk: a framework for providing self-integrating context-aware services. *Knowledge-Based Systems*, 11(1):3 – 13.
- [Dockhorn Costa, 2007] Dockhorn Costa, P. (2007). *Architectural support for context-aware applications: from context models to services platforms*. University of Twente.
- [Dressler et al., 2009] Dressler, F., Dietrich, I., German, R., and Kruger, B. (2009). A rule-based system for programming self-organized sensor and actor networks. *Computer Networks*, 53(10):1737 – 1750. Autonomic and Self-Organising Systems.
- [Dumas et al., 2002] Dumas, M., Governatori, G., Ter Hofstede, A. H., and Oaks, P. (2002). A formal approach to negotiating agents development. *Electronic commerce research and applications*, 1(2):193–207.
- [Eiter et al., 2014] Eiter, T., Fink, M., Schuller, P., and Weinzierl, A. (2014). Finding explanations of inconsistency in multi-context systems. *Artificial Intelligence*, 216:233 – 274.
- [Eiter et al., 2011] Eiter, T., Ianni, G., Lukasiewicz, T., and Schindlauer, R. (2011). Well-founded semantics for description logic programs in the semantic web. *ACM Transactions on Computational Logic (TOCL)*, 12(2):11.
- [Ejigu et al., 2007] Ejigu, D., Scuturici, M., and Brunie, L. (2007). An ontology-based approach to context modeling and reasoning in pervasive computing. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 14–19.

- [Eker et al., 2003] Eker, S., Meseguer, J., and Sridharanarayanan, A. (2003). The maude ltl model checker and its implementation. In Ball, T. and Rajamani, S., editors, *Model Checking Software*, volume 2648 of *Lecture Notes in Computer Science*, pages 230–234. Springer Berlin Heidelberg.
- [Elrod et al., 1993] Elrod, S., Hall, G., Costanza, R., Dixon, M., and Des Rivières, J. (1993). Responsive office environments. *Commun. ACM*, 36(7):84–85.
- [Esposito et al., 2008] Esposito, A., Tarricone, L., Zappatore, M., Catarinucci, L., Colella, R., and DiBari, A. (2008). A framework for context-aware home-health monitoring. In Sandnes, F., Zhang, Y., Rong, C., Yang, L., and Ma, J., editors, *Ubiquitous Intelligence and Computing*, volume 5061 of *Lecture Notes in Computer Science*, pages 119–130. Springer Berlin Heidelberg.
- [Faruqui, 2012] Faruqui, M. R. U. (2012). Scalable reasoning over large ontologies. Master’s thesis, Saint Francis Xavier University.
- [Fensel et al., 2001] Fensel, D., Van Harmelen, F., Horrocks, I., McGuinness, D. L., and Patel-Schneider, P. F. (2001). Oil: An ontology infrastructure for the semantic web. *Intelligent Systems, IEEE*, 16(2):38–45.
- [Fickas et al., 1997] Fickas, S., Kortuem, G., and Segall, Z. (1997). Software organization for dynamic and adaptable wearable systems. In *Wearable Computers, 1997. Digest of Papers., First International Symposium on*, pages 56–63.
- [Friedman, 2003] Friedman, E. (2003). *Jess in Action: Rule-based Systems in Java*. Manning Publications Co., Greenwich, CT, USA.
- [Fu and Fu, 2015] Fu, J. and Fu, Y. (2015). An adaptive multi-agent system for cost collaborative management in supply chains. *Engineering Applications of Artificial Intelligence*, 44:91 – 100.

- [Gammie and van der Meyden, 2004] Gammie, P. and van der Meyden, R. (2004). *Computer Aided Verification: 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004. Proceedings*, chapter MCK: Model Checking the Logic of Knowledge, pages 479–483. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [García and Simari, 2004] García, A. J. and Simari, G. R. (2004). Defeasible logic programming: An argumentative approach. *Theory and practice of logic programming*, 4(1+ 2):95–138.
- [Ghidini and Giunchiglia, 2001] Ghidini, C. and Giunchiglia, F. (2001). Local models semantics, or contextual reasoning=locality+compatibility. *Artificial Intelligence*, 127(2):221 – 259.
- [Gómez et al., 2006] Gómez, S. A., Chesnevar, C. I., and Simari, G. R. (2006). An approach to handling inconsistent ontology definitions based on the translation of description logics into defeasible logic programming. In *XII Congreso Argentino de Ciencias de la Computación*, pages 1185–1196.
- [Gómez et al., 2007] Gómez, S. A., Chesnevar, C. I., and Simari, G. R. (2007). Inconsistent ontology handling by translating description logics into defeasible logic programming. *Revista Iberoamericana de Inteligencia Artificial*, 11(35):11–22.
- [Gómez et al., 2010] Gómez, S. A., Chesnevar, C. I., and Simari, G. R. (2010). A defeasible logic programming approach to the integration of rules and ontologies. *Journal of Computer Science & Technology*, 10:74–80.
- [Governatori, 2005] Governatori, G. (2005). Representing business contracts in ruleml. *International Journal of Cooperative Information Systems*, 14(02n03):181–216.
- [Governatori et al., 2004] Governatori, G., Maher, M. J., Antoniou, G., and Billington, D. (2004). Argumentation semantics for defeasible logic. *Journal of Logic and Computation*, 14(5):675–702.

- [Governatori and Pham Hoang, 2005] Governatori, G. and Pham Hoang, D. (2005). Dr-contract: an architecture for e-contracts in defeasible logic. In *2nd EDOC Workshop on Contract Architectures and Languages (CoALA 2005)*, pages 1–9. IEEE.
- [Governatori and Rotolo, 2004] Governatori, G. and Rotolo, A. (2004). Defeasible logic: Agency, intention and obligation. In *Deontic logic in computer science*, pages 114–128. Springer.
- [Governatori et al., 2007] Governatori, G., Rotolo, A., Riveret, R., Palmirani, M., and Sartor, G. (2007). Variants of temporal defeasible logics for modelling norm modifications. In *Proceedings of the 11th international conference on Artificial intelligence and law*, pages 155–159. ACM.
- [Governatori et al., 2005] Governatori, G., Rotolo, A., and Sartor, G. (2005). Temporalised normative positions in defeasible logic. In *Proceedings of the 10th international conference on Artificial intelligence and law*, pages 25–34. ACM.
- [Grosof, 1997] Grosf, B. N. (1997). Prioritized conflict handling for logic programs. In *ILPS*, volume 97, pages 197–211.
- [Grosof et al., 2003] Grosf, B. N., Horrocks, I., Volz, R., and Decker, S. (2003). Description logic programs: Combining logic programs with description logic. In *Proceedings of the 12th International Conference on World Wide Web, WWW '03*, pages 48–57, New York, NY, USA. ACM.
- [Grosof et al., 1999] Grosf, B. N., Labrou, Y., and Chan, H. Y. (1999). A declarative approach to business rules in contracts: courteous logic programs in xml. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 68–77. ACM.
- [Grosof and Poon, 2003] Grosf, B. N. and Poon, T. C. (2003). Sweetdeal: representing agent contracts with exceptions using xml rules, ontologies, and process descriptions. In *Proceedings of the 12th international conference on World Wide Web*, pages 340–349. ACM.

- [Gruber,] Gruber, T. What is an ontology? <http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>. Accessed: 2016-02-04.
- [Haarslev and Möller, 2001] Haarslev, V. and Möller, R. (2001). Racer system description. In Gor, R., Leitsch, A., and Nipkow, T., editors, *Automated Reasoning*, volume 2083 of *Lecture Notes in Computer Science*, pages 701–705. Springer Berlin Heidelberg.
- [Haase et al., 2008] Haase, P., Lewen, H., Studer, R., Tran, D. T., Erdmann, M., d’Aquin, M., and Motta, E. (2008). The neon ontology engineering toolkit. In *Jeff Korn, editor, WWW 2008 Developers Track*.
- [Hayes-Roth, 1985] Hayes-Roth, F. (1985). Rule-based systems. *Commun. ACM*, 28(9):921–932.
- [Hedman, 2004] Hedman, S. (2004). *A first course in logic: an introduction to model theory, proof theory, computability, and complexity*. Number 9. Oxford University Press Oxford.
- [Helal et al., 2003] Helal, S., Winkler, B., Lee, C., Kaddoura, Y., Ran, L., Giraldo, C., Kuchibhotla, S., and Mann, W. (2003). Enabling location-aware pervasive computing applications for the elderly. In *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on*, pages 531–536.
- [Hendler and McGuinness, 2000] Hendler, J. and McGuinness, D. L. (2000). The darpa agent markup language. *IEEE Intelligent systems*, 15(6):67–73.
- [Henricksen et al., 2002] Henricksen, K., Indulska, J., and Rakotonirainy, A. (2002). Modeling context information in pervasive computing systems. In Mattern, F. and Naghshineh, M., editors, *Pervasive Computing*, volume 2414 of *Lecture Notes in Computer Science*, pages 167–180. Springer Berlin Heidelberg.

- [Hintikka, 1962] Hintikka, J. (1962). *Knowledge and Belief*. Ithaca, N.Y., Cornell University Press.
- [Hitzler et al., 2009] Hitzler, P., Krötzsch, M., Parsia, B., and Rudolph, S. (2009). Owl 2 web ontology language primer. *W3C recommendation*, 27(1):80–85.
- [Holzmann, 1997] Holzmann, G. J. (1997). The model checker spin. *IEEE Transactions on software engineering*, 23(5):page 279.
- [Hong et al., 2009] Hong, J., Suh, E.-H., Kim, J., and Kim, S. (2009). Context-aware system for proactive personalized service based on context history. *Expert Systems with Applications*, 36(4):7448 – 7457.
- [Hong and Cho, 2008] Hong, M.-w. and Cho, D.-j. (2008). Ontology context model for context-aware learning service in ubiquitous learning environments. *International Journal of Computers*, 2(3):172–178.
- [Horridge and Bechhofer, 2009] Horridge, M. and Bechhofer, S. (2009). The OWL API: A java API for working with OWL 2 Ontologies. In *6th OWL Experienced and Directions Workshop (OWLED)*, volume 529, pages 49–58.
- [Horridge and Bechhofer, 2011] Horridge, M. and Bechhofer, S. (2011). The owl api: A java api for owl ontologies. *Semantic Web*, 2(1):11–21.
- [Horrocks et al., 2002] Horrocks, I. et al. (2002). Daml+oil: a description logic for the semantic web. *IEEE Data Eng. Bull.*, 25(1):4–9.
- [Horrocks et al., 2006] Horrocks, I., Kutz, O., and Sattler, U. (2006). The even more irresistible sroiq. *KR*, 6:57–67.
- [Horrocks et al., 2004] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosz, B., and Dean, M. (2004). SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission.

- [Hull et al., 1997] Hull, R., Neaves, P., and Bedford-Roberts, J. (1997). Towards situated computing. In *Wearable Computers, 1997. Digest of Papers., First International Symposium on*, pages 146–153.
- [Hustadt and Schmidt, 2000] Hustadt, U. and Schmidt, R. A. (2000). Issues of decidability for description logics in the framework of resolution. In *Automated Deduction in Classical and Non-Classical Logics*, pages 191–205. Springer.
- [Hustadt et al., 2004] Hustadt, U., Schmidt, R. A., and Georgieva, L. (2004). A survey of decidable first-order fragments and description logics. *Journal of Relational Methods in Computer Science*, 1:(3) 251–276.
- [Huth and Ryan, 2004] Huth, M. and Ryan, M. (2004). *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge University Press.
- [Intille et al., 2002] Intille, S. S., Larson, K., and Kukla, C. (2002). Just-in-time context-sensitive questioning for preventative health care. In *Proceedings of the AAAI 2002 Workshop on Automation as Caregiver: The Role of Intelligent Technology in Elder Care, AAAI Technical Report WS-02-02*. AAAI Press, Menlo Park, CA, 2002.
- [Jago, 2006] Jago, M. (2006). *Logics for resource-bounded agents*. PhD thesis, University of Nottingham.
- [Jago, 2009] Jago, M. (2009). Epistemic logic for rule-based agents. *Journal of Logic, Language and Information*, 18(1):131–158.
- [Kakas et al., 1992] Kakas, A. C., Kowalski, R. A., and Toni, F. (1992). Abductive logic programming. *Journal of logic and computation*, 2(6):719–770.
- [Kindberg and Barton, 2001] Kindberg, T. and Barton, J. (2001). A web-based nomadic computing system. *Computer Networks*, 35(4):443 – 456. Pervasive Computing.

- [Knublauch et al., 2004] Knublauch, H., Fergerson, R. W., Noy, N. F., and Musen, M. A. (2004). The protégé owl plugin: An open development environment for semantic web applications. In *The Semantic Web–ISWC 2004*, pages 229–243. Springer.
- [Koch and Rahwan, 2004] Koch, F. and Rahwan, I. (2004). Classification of agents-based mobile assistants. In *Proceedings of the AAMAS workshop on agents for ubiquitous computing (UbiA-gents), New York, USA, Jul 2004*.
- [Kravari et al., 2010] Kravari, K., Kastori, G.-E., Bassiliades, N., and Governatori, G. (2010). A contract agreement policy-based workflow methodology for agents interacting in the semantic web. In *Semantic Web Rules*, pages 225–239. Springer.
- [Krötzsch et al., 2011] Krötzsch, M., Maier, F., Krisnadhi, A., and Hitzler, P. (2011). A better uncle for owl: Nominal schemas for integrating rules and ontologies. In *Proceedings of the 20th international conference on World wide web*, pages 645–654. ACM.
- [Kuba, 2012] Kuba, M. (2012). Owl 2 and swrl tutorial. From <http://dior.ics.muni.cz/~makub/owl/#ontology.htm>. retrieved on 29 December, 2015.
- [Kwon et al., 2005] Kwon, O., Choi, S., and Park, G. (2005). Nama: a context-aware multi-agent based web service approach to proactive need identification for personalized reminder systems. *Expert Systems with Applications*, 29(1):17 – 32.
- [Kwon and Sadeh, 2004] Kwon, O. B. and Sadeh, N. (2004). Applying case-based reasoning and multi-agent intelligent system to context-aware comparative shopping. *Decision Support Systems*, 37(2):199 – 213.
- [Lam, 2012] Lam, H. P. (2012). On the derivability of defeasible logic, school of information technology and electrical engineering. The University of Queensland, Queensland.
- [Leijdekkers and Gay, 2006] Leijdekkers, P. and Gay, V. (2006). Personal heart monitoring system using smart phones to detect life threatening arrhythmias. In *Computer-*

Based Medical Systems, 2006. CBMS 2006. 19th IEEE International Symposium on, pages 157–164.

[Lesser et al., 1999] Lesser, V., Atighetchi, M., Benyo, B., Horling, B., Raja, A., Wagner, T., and Xuan, P. (1999). The intelligent home testbed. *environment*, 2:15.

[Levy and Rousset, 1998] Levy, A. Y. and Rousset, M.-C. (1998). Combining horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1-2):165–209.

[Liebig and Noppens, 2004] Liebig, T. and Noppens, O. (2004). Ontotrack: Combining browsing and editing with reasoning and explaining for owl lite ontologies. In *Proceedings of the 3rd International Semantic Web Conference ISWC 2004. Hiroshima, Japan*, pages 8–11. Springer.

[LIRIS, 2010] LIRIS, L. B. R. (2010). *A dynamic trust-based context-aware secure authentication framework for pervasive computing environments*. PhD thesis, Telecom & Management SudParis (France).

[Lomuscio et al., 2009] Lomuscio, A., Qu, H., and Raimondi, F. (2009). *Computer Aided Verification: 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, chapter MCMAS: A Model Checker for the Verification of Multi-Agent Systems, pages 682–688. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Lu and Sadiq, 2007] Lu, R. and Sadiq, S. (2007). A survey of comparative business process modeling approaches. In *Business information systems*, pages 82–94. Springer.

[Maher, 2001] Maher, M. J. (2001). Propositional defeasible logic has linear complexity. *Theory and Practice of Logic Programming*, 1(06):691–711.

[Maher, 2002] Maher, M. J. (2002). A model-theoretic semantics for defeasible logic. *Workshop on Paraconsistent Computational Logic*, pages 67–80.

- [Maher et al., 2001] Maher, M. J., Rock, A., Antoniou, G., Billington, D., and Miller, T. (2001). Efficient defeasible reasoning systems. *International Journal on Artificial Intelligence Tools*, 10(04):483–501.
- [McCarthy, 1987] McCarthy, J. (1987). Circumscription: a form of nonmonotonic reasoning. *Artificial Intelligence*, 13:27–39.
- [McDermott and Doyle, 1980] McDermott, D. and Doyle, J. (1980). Non-monotonic logic i. *Artificial intelligence*, 13(1):41–72.
- [Moore, 1985] Moore, R. C. (1985). Semantical considerations on nonmonotonic logic. *Artificial intelligence*, 25(1):75–94.
- [Motik et al., 2005] Motik, B., Sattler, U., and Studer, R. (2005). Query answering for OWL-DL with rules. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 3:41–60.
- [Motik et al., 2009] Motik, B., Shearer, R., and Horrocks, I. (2009). Hypertableau reasoning for description logics. *Journal of Artificial Intelligence Research*, 36(1):165–228.
- [Motik and Studer, 2005] Motik, B. and Studer, R. (2005). Kaon2-a scalable reasoning tool for the semantic web. In *Proceedings of the 2nd European Semantic Web Conference (ESWC'05), Heraklion, Greece*, volume 17.
- [Mozer, 1999] Mozer, M. (1999). An intelligent environment must be adaptive. *Intelligent Systems and their Applications, IEEE*, 14(2):11–13.
- [Nabih et al., 2011] Nabih, A. K., Gomaa, M. M., Osman, H. S., Aly, G. M., Azid, S. I., Kumar, S., Colace, F., De Santo, M., Abdullah, M., and Poh, L. M. (2011). Modeling, simulation, and control of smart homes using petri nets. *International Journal of Smart Home*, 5(3):1–14.

- [Nga, 2011] Nga, N. H. (2011). *Reasoning about Resource-bounded Multi-agent Systems*. PhD thesis, University Of Nottingham.
- [Nute, 2003] Nute, D. (2003). Defeasible logic. In *Web Knowledge Management and Decision Support*, pages 151–169. Springer.
- [O'Connor and Das, 2012] O'Connor, M. J. and Das, A. (2012). A pair of owl 2 rl reasoners. In *Pavel Klinov and Matthew Horridge, editors, OWLED, volume 849 of CEUR Workshop Proceedings. CEUR-WS.org*.
- [Pascoe, 1998] Pascoe, J. (1998). Adding generic contextual capabilities to wearable computers. In *Wearable Computers, 1998. Digest of Papers. Second International Symposium on*, pages 92–99. IEEE.
- [Patel-Schneider, 1987] Patel-Schneider, P. F. (1987). Decidable, logic-based knowledge representation. *PhD Thesis, Department of Computer Science, University of Toronto, Toronto*.
- [Paul, 1993] Paul, G. (1993). Approaches to abductive reasoning: an overview. *Artificial intelligence review*, 7(2):109–152.
- [Perera et al., 2014] Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014). Context aware computing for the internet of things: A survey. *Communications Surveys Tutorials, IEEE*, 16(1):414–454.
- [Pnueli, 1977] Pnueli, A. (1977). The temporal logic of programs. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 46–57.
- [Priyantha et al., 2000] Priyantha, N. B., Chakraborty, A., and Balakrishnan, H. (2000). The cricket location-support system. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, MobiCom '00*, pages 32–43, New York, NY, USA. ACM.

- [Protégé, 2011] Protégé (2011). The Protégé ontology editor and knowledge-base framework (Version 4.1). <http://protege.stanford.edu/>. Accessed: 2016-08-15.
- [Qin et al., 2007] Qin, W., Shi, Y., and Suo, Y. (2007). Ontology-based context-aware middleware for smart spaces. *Tsinghua Science & Technology*, 12(6):707–713.
- [Rakib, 2011] Rakib, A. (2011). *Verifying Requirements For Resource-Bounded Agents*. PhD thesis, University Of Nottingham.
- [Rakib, 2012] Rakib, A. (2012). Formal approaches to modelling and verifying resource-bounded agents-state of the art and future prospects. *Journal of Information Technology & Software Engineering*, 2(4).
- [Rakib and Faruqui, 2013] Rakib, A. and Faruqui, R. (2013). A formal approach to modelling and verifying resource-bounded context-aware agents. In Vinh, P., Hung, N., Tung, N., and Suzuki, J., editors, *Context-Aware Systems and Applications*, volume 109 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 86–96. Springer Berlin Heidelberg.
- [Rakib et al., 2012] Rakib, A., Faruqui, R., and MacCaull, W. (2012). Verifying resource requirements for ontology-driven rule-based agents. In Lukasiewicz, T. and Sali, A., editors, *Foundations of Information and Knowledge Systems*, volume 7153 of *Lecture Notes in Computer Science*, pages 312–331. Springer Berlin Heidelberg.
- [Rakib and Haque, 2014] Rakib, A. and Haque, H. (2014). A logic for context-aware non-monotonic reasoning agents. In Gelbukh, A., Espinoza, F., and Galicia-Haro, S., editors, *Human-Inspired Computing and Its Applications*, volume 8856 of *Lecture Notes in Computer Science*, pages 453–471. Springer International Publishing.
- [Rakib et al., 2014] Rakib, A., Ul Haque, H., and Faruqui, R. (2014). A temporal description logic for resource-bounded rule-based context-aware agents. In Vinh, P. C., Alagar, V., Vassev, E., and Khare, A., editors, *Context-Aware Systems and Applications*, vol-

- ume 128 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 3–14. Springer International Publishing.
- [Reiter, 1980] Reiter, R. (1980). A logic for default reasoning. *Artificial intelligence*, 13(1):81–132.
- [Rekimoto et al., 1998] Rekimoto, J., Ayatsuka, Y., and Hayashi, K. (1998). Augmentable reality: situated communication through physical and digital spaces. In *Wearable Computers, 1998. Digest of Papers. Second International Symposium on*, pages 68–75.
- [Reynolds, 2001] Reynolds, M. (2001). An axiomatization of full computation tree logic. *Journal of Symbolic Logic*, 66(3):1011–1057.
- [Robinson, 1965] Robinson, J. A. (1965). A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41.
- [Rosati, 2006] Rosati, R. (2006). DL+log: Tight integration of description logics and disjunctive datalog. In *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 68–78. AAAI Press.
- [Rudolph, 2011] Rudolph, S. (2011). Foundations of description logics. In *Reasoning Web. Semantic Technologies for the Web of Data*, pages 76–136. Springer.
- [Ryan et al., 1999] Ryan, N., Pascoe, J., and Morse, D. (1999). Enhanced reality fieldwork: the context aware archaeological assistant. *Bar International Series*, 750:269–274.
- [Schilit et al., 1994] Schilit, B., Adams, N., and Want, R. (1994). Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pages 85–90.
- [Schilit and Theimer, 1994] Schilit, B. and Theimer, M. (1994). Disseminating active map information to mobile hosts. *Network, IEEE*, 8(5):22–32.

- [Schmidt et al., 1999] Schmidt, A., Beigl, M., and Gellersen, H.-W. (1999). There is more to context than location. *Computers & Graphics*, 23(6):893 – 901.
- [Schmidt-Schauß and Smolka, 1991] Schmidt-Schauß, M. and Smolka, G. (1991). Attributive concept descriptions with complements. *Artificial intelligence*, 48(1):1–26.
- [Shadbolt et al., 2006] Shadbolt, N., Hall, W., and Berners-Lee, T. (2006). The semantic web revisited. *IEEE Intelligent Systems* 21(3), 21(3):96–101.
- [Sirin et al., 2007] Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53.
- [Skylogiannis et al., 2007] Skylogiannis, T., Antoniou, G., Bassiliades, N., Governatori, G., and Bikakis, A. (2007). Dr-negotiate—a system for automated agent negotiation with defeasible logic-based strategies. *Data & Knowledge Engineering*, 63(2):362–380.
- [Smullyan, 1995] Smullyan, R. M. (1995). *First-order logic*. ISBN-13: 080-0759683703, Dover Publications, Courier Corporation.
- [Stanford, 2002] Stanford, V. (2002). Using pervasive computing to deliver elder care. *Pervasive Computing, IEEE*, 1(1):10–13.
- [Stirling, 2012] Stirling, C. (2012). Bisimulation and logic. *Sangiorgi and Rutten [24, Chapter 4]*, pages 173–196.
- [Strang and Linnhoff-Popien, 2004] Strang, T. and Linnhoff-Popien, C. (2004). A context modeling survey. In *Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England, September*.

- [Strang et al., 2003] Strang, T., Linnhoff-Popien, C., and Frank, K. (2003). Cool: A context ontology language to enable contextual interoperability. In *Distributed applications and interoperable systems*, pages 236–247. Springer.
- [Ter Horst, 2005] Ter Horst, H. J. (2005). Completeness, decidability and complexity of entailment for rdf schema and a semantic extension involving the owl vocabulary. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2):79–115.
- [Terfloth et al., 2006] Terfloth, K., Wittenburg, G., and Schiller, J. (2006). Facts - a rule-based middleware architecture for wireless sensor networks. In *Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on*, pages 1–8.
- [Tsarkov and Horrocks, 2006] Tsarkov, D. and Horrocks, I. (2006). Fact++ description logic reasoner: System description. In Furbach, U. and Shankar, N., editors, *Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*, pages 292–297. Springer Berlin Heidelberg.
- [Van der Hoek and Wooldridge, 2002] Van der Hoek, W. and Wooldridge, M. (2002). *Model Checking Software: 9th International SPIN Workshop Grenoble, France, April 11–13, 2002 Proceedings*, chapter Model Checking Knowledge and Time, pages 95–111. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Van Harmelen and McGuinness, 2004] Van Harmelen, F. and McGuinness, D. L. (2004). Owl web ontology language overview. *World Wide Web Consortium (W3C) Recommendation*.
- [Vilasrao and Bhaskar, 2012] Vilasrao, V. V. and Bhaskar, P. (2012). Mining web data based on ontology and swrl. *International Journal of Information and Education Technology*, 2(5).
- [Walter Sinnott-Armstrong, 1986] Walter Sinnott-Armstrong, James Moor, R. F. (1986). A defense of modus ponens. *The Journal of Philosophy*, 83(5):296–300.

- [Wang et al., 2004a] Wang, K., Billington, D., Blee, J., and Antoniou, G. (2004a). *Combining Description Logic and Defeasible Logic for the Semantic Web*, pages 170–181. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Wang et al., 2004b] Wang, X., Zhang, D. Q., Gu, T., and Pung, H. (2004b). Ontology based context modeling and reasoning using owl. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 18–22.
- [Want et al., 1992] Want, R., Hopper, A., Falcão, V., and Gibbons, J. (1992). The active badge location system. *ACM Trans. Inf. Syst.*, 10(1):91–102.
- [Weiser, 1999] Weiser, M. (1999). The computer for the 21st century. *ACM SIGMOBILE Mobile Computing and Communications Review— Special issue dedicated to Mark Weiser*, 3(3):3–11.
- [Wolper, 1983] Wolper, P. (1983). Temporal logic can be more expressive. *Information and Control*, 56(1 - 2):72 – 99.
- [Wooldridge et al., 2002] Wooldridge, M., Fisher, M., Huget, M.-P., and Parsons, S. (2002). Model checking multi-agent systems with mable. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 2, AAMAS '02*, pages 952–959, New York, NY, USA. ACM.
- [Yang et al., 2003] Yang, G., Kifer, M., and Zhao, C. (2003). *Flora-2: A rule-based knowledge representation and inference infrastructure for the semantic web*. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pages 671–688. Springer.
- [Zhang et al., 2005] Zhang, D., Gu, T., and Wang, X. (2005). Enabling context-aware smart home with semantic web technologies. *International Journal of Human-friendly Welfare Robotic Systems*, 6(4):12–20.

Appendix A

A set of rules for Smart Environment Case Study

A.1 Patient Care Agent

Initial Facts: $\text{Person}(\text{'Philip'}), \text{hasPatientID}(\text{'Philip'}, \text{'P001'}), \text{PatientID}(\text{'P001'})$

Rules:

1. $\langle 1 : \text{Person}(\text{'?p'}), \text{hasPatientID}(\text{'?p'}, \text{'?pid'}), \text{PatientID}(\text{'?pid'}) \rightarrow \text{Patient}(\text{'?p'}) \rangle$
2. $\langle 2 : \text{Patient}(\text{'?p'}), \text{hasBloodPressure}(\text{'?p'}, \text{'Normal'}) \rightarrow \text{Not}(\text{hasSituation}(\text{'?p'}, \text{'Emergency'})) \rangle$
3. $\langle 2 : \text{Patient}(\text{'?p'}), \text{hasBloodPressure}(\text{'?p'}, \text{'Prehypertension'}) \rightarrow \text{Not}(\text{hasSituation}(\text{'?p'}, \text{'Emergency'})) \rangle$
4. $\langle 4 : \text{Patient}(\text{'?p'}), \text{hasBloodPressure}(\text{'?p'}, \text{'Stage1hypertension'}) \rightarrow \text{hasSituation}(\text{'?p'}, \text{'OnCall'}) \rangle$
5. $\langle 5 : \text{Patient}(\text{'?p'}), \text{hasBloodPressure}(\text{'?p'}, \text{'Stage2hypertension'}) \rightarrow \text{hasSituation}(\text{'?p'}, \text{'Emergency'}) \rangle$

6. $\langle 5 : Patient(?p), hasBloodPressure(?p, 'Hypotension) \rightarrow$
 $hasSituation(?p, 'Emergency) \rangle$
7. $\langle 2 : Patient(?p), hasDBCategory(?p, 'Controlled) \rightarrow$
 $Not(hasSituation(?p, 'Emergency)) \rangle$
8. $\langle 2 : Patient(?p), hasDBCategory(?p, 'EstablishedDiabetes) \rightarrow$
 $Not(hasSituation(?p, 'Emergency)) \rangle$
9. $\langle 4 : Patient(?p), hasDBCategory(?p, 'Type2Diabetes) \rightarrow$
 $hasSituation(?p, 'OnCall) \rangle$
10. $\langle 5 : Patient(?p), hasDBCategory(?p, 'Hyperglycaemia) \rightarrow$
 $hasSituation(?p, 'Emergency) \rangle$
11. $\langle 5 : Patient(?p), hasDBCategory(?p, 'Hypoglycaemia) \rightarrow$
 $hasSituation(?p, 'Emergency) \rangle$
12. $\langle 5 : Patient(?p), hasFever(?p, 'Hypothermia) \rightarrow hasSituation(?p, 'Emergency)$
 \rangle
13. $\langle 2 : Patient(?p), hasFever(?p, 'Normal) \rightarrow Not(hasSituation(?p, 'Emergency))$
 \rangle
14. $\langle 2 : Patient(?p), hasFever(?p, 'High) \rightarrow Not(hasSituation(?p, 'Emergency)) \rangle$
15. $\langle 4 : Patient(?p), hasFever(?p, 'Hyperthermia) \rightarrow hasSituation(?p, 'OnCall) \rangle$
16. $\langle 5 : Patient(?p), hasFever(?p, 'Hyperpyrexia) \rightarrow hasSituation(?p, 'Emergency)$
 \rangle
17. $\langle 5 : Patient(?p), hasPulseRate(?p, 'Abnormal) \rightarrow hasSituation(?p, 'Emergency)$
 \rangle
18. $\langle 2 : Patient(?p), hasPulseRate(?p, 'Normal) \rightarrow$
 $Not(hasSituation(?p, 'Emergency)) \rangle$

19. $\langle 4 : Patient(?p), hasBloodPressure(?p, 'Stage1hypertension),$
 $hasDBCcategory(?p, 'EstablishedDiabetes), hasFever(?p, 'High) \rightarrow$
 $hasSituation(?p, 'OnCall) \rangle$
20. $\langle 3 : Patient(?p), hasBloodPressure(?p, 'Prehypertension),$
 $hasDBCcategory(?p, 'Controlled), hasFever(?p, 'Normal) \rightarrow$
 $Not(hasSituation(?p, 'OnCall)) \rangle$
21. $\langle 2 : Patient(?p), hasBloodPressure(?p, 'Prehypertension),$
 $hasDBCcategory(?p, 'EstablishedDiabetes), hasFever(?p, 'Normal) \rightarrow$
 $Not(hasSituation(?p, 'Emergency)) \rangle$
22. $\langle 5 : Patient(?p), hasFever(?p, 'Hyperpyrexia),$
 $hasDBCcategory(?p, 'EstablishedDiabetes) \rightarrow hasSituation(?p, 'Emergency) \rangle$
23. $\langle 4 : Patient(?p), hasFever(?p, 'Normal),$
 $hasDBCcategory(?p, 'Type2Diabetes) \rightarrow hasSituation(?p, 'OnCall) \rangle$
24. $\langle 5 : Patient(?p), hasDBCcategory(?p, 'Type2Diabetes), hasPulseRate(?p, 'Abnormal)$
 $\rightarrow hasSituation(?p, 'Emergency) \rangle$
25. $\langle 5 : Patient(?p), hasBloodPressure(?p, 'Prehypertension),$
 $hasDBCcategory(?p, 'Hyperglycaemia), hasPulseRate(?p, 'Abnormal) \rightarrow$
 $hasSituation(?p, 'Emergency) \rangle$
26. $\langle 5 : Patient(?p), hasFever(?p, 'Hyperpyrexia), hasPulseRate(?p, 'Abnormal)$
 $\rightarrow hasSituation(?p, 'Emergency) \rangle$
27. $\langle 5 : Patient(?p), hasBloodPressure(?p, 'Stage2hypertension),$
 $hasDBCcategory(?p, 'Type2Diabetes) \rightarrow hasSituation(?p, 'Emergency) \rangle$
28. $\langle 5 : Patient(?p), hasBloodPressure(?p, 'Stage2hypertension),$
 $hasDBCcategory(?p, 'Hyperglycaemia), hasFever(?p, 'Hyperpyrexia),$
 $hasPulseRate(?p, 'Abnormal) \rightarrow hasSituation(?p, 'Emergency) \rangle$

29. $< 6 : TELL(2, 1, hasBloodPressure(?p, 'Normal')) \rightarrow$
 $hasBloodPressure(?p, 'Normal) >$
30. $< 6 : TELL(2, 1, hasBloodPressure(?p, 'Prehypertension')) \rightarrow$
 $hasBloodPressure(?p, 'Prehypertension) >$
31. $< 6 : TELL(2, 1, hasBloodPressure(?p, 'Stage1hypertension')) \rightarrow$
 $hasBloodPressure(?p, 'Stage1hypertension) >$
32. $< 6 : TELL(2, 1, hasBloodPressure(?p, 'Stage2hypertension')) \rightarrow$
 $hasBloodPressure(?p, 'Stage2hypertension) >$
33. $< 6 : TELL(2, 1, hasBloodPressure(?p, 'Hypotension')) \rightarrow$
 $hasBloodPressure(?p, 'Hypotension) >$
34. $< 6 : TELL(3, 1, hasDBCcategory(?p, 'Controlled')) \rightarrow$
 $hasDBCcategory(?p, 'Controlled) >$
35. $< 6 : TELL(3, 1, hasDBCcategory(?p, 'EstablishedDiabetes')) \rightarrow$
 $hasDBCcategory(?p, 'EstablishedDiabetes) >$
36. $< 6 : TELL(3, 1, hasDBCcategory(?p, 'Type2Diabetes')) \rightarrow$
 $hasDBCcategory(?p, 'Type2Diabetes) >$
37. $< 6 : TELL(3, 1, hasDBCcategory(?p, 'Hyperglycaemia')) \rightarrow$
 $hasDBCcategory(?p, 'Hyperglycaemia) >$
38. $< 6 : TELL(3, 1, hasDBCcategory(?p, 'Hypoglycaemia')) \rightarrow$
 $hasDBCcategory(?p, 'Hypoglycaemia) >$
39. $< 6 : TELL(4, 1, hasFever(?p, 'Hypothermia')) \rightarrow hasFever(?p, 'Hypothermia)$
 $>$
40. $< 6 : TELL(4, 1, hasFever(?p, 'Normal')) \rightarrow hasFever(?p, 'Normal) >$

41. $\langle 6 : TELL(4, 1, hasFever(?p, 'High)) \rightarrow hasFever(?p, 'High) \rangle$
42. $\langle 6 : TELL(4, 1, hasFever(?p, 'Hyperthermia)) \rightarrow hasFever(?p, 'Hyperthermia) \rangle$
43. $\langle 6 : TELL(4, 1, hasFever(?p, 'Hyperpyrexia)) \rightarrow hasFever(?p, 'Hyperpyrexia) \rangle$
44. $\langle 6 : TELL(5, 1, hasPulseRate(?p, 'Normal)) \rightarrow hasPulseRate(?p, 'Normal) \rangle$
45. $\langle 6 : TELL(5, 1, hasPulseRate(?p, 'Abnormal)) \rightarrow hasPulseRate(?p, 'Abnormal) \rangle$

A.2 Blood Pressure Agent

Initial Facts: Person('Philip) , SystolicBP('134), DiastolicBP('88),
 hasSystolicBP('Philip, '134), hasDiastolicBP('Philip, '88), greaterThan('134,'120),
 lessThan('134,'140), greaterThan('88,'80), lessThan('88,'90)

Rules:

1. $\langle 1 : Person(?p), SystolicBP(?sbp), DiastolicBP(?dbp), hasSystolicBP(?p, ?sbp), hasDiastolicBP(?p, ?dbp), greaterThan(?sbp, '90), lessThan(?sbp, '120), greaterThan(?dbp, '60), lessThan(?dbp, '80) \rightarrow hasBloodPressure(?p, 'Normal) \rangle$
2. $\langle 1 : Person(?p), SystolicBP(?sbp), DiastolicBP(?dbp), hasSystolicBP(?p, ?sbp), hasDiastolicBP(?p, ?dbp), greaterThan(?sbp, '120), lessThan(?sbp, '140), greaterThan(?dbp, '80), lessThan(?dbp, '90) \rightarrow hasBloodPressure(?p, 'Prehypertension) \rangle$
3. $\langle 1 : Person(?p), SystolicBP(?sbp), DiastolicBP(?dbp), hasSystolicBP(?p, ?sbp), hasDiastolicBP(?p, ?dbp), greaterThan(?sbp, '140), lessThan(?sbp, '160), greaterThan(?dbp, '90), lessThan(?dbp, '100) \rightarrow hasBloodPressure(?p, 'Stage1hypertension) \rangle$

4. $< 1 : \text{Person}(?p), \text{SystolicBP}(?sbp), \text{DiastolicBP}(?dbp), \text{hasSystolicBP}(?p, ?sbp),$
 $\text{hasDiastolicBP}(?p, ?dbp), \text{greaterThan}(?sbp, '160), \text{greaterThan}(?dbp, '100)$
 $\rightarrow \text{hasBloodPressure}(?p, ' \text{Stage2hypertension}) >$
5. $< 1 : \text{Person}(?p), \text{SystolicBP}(?sbp), \text{DiastolicBP}(?dbp), \text{hasSystolicBP}(?p, ?sbp),$
 $\text{hasDiastolicBP}(?p, ?dbp), \text{lessThan}(?sbp, '90), \text{lessThan}(?dbp, '60)$
 $\rightarrow \text{hasBloodPressure}(?p, ' \text{Hypotension}) >$
6. $< 2 : \text{hasBloodPressure}(?p, ' \text{Normal}) \rightarrow$
 $\text{TELL}(2, 1, \text{hasBloodPressure}(?p, ' \text{Normal})) >$
7. $< 2 : \text{hasBloodPressure}(?p, ' \text{Prehypertension}) \rightarrow$
 $\text{TELL}(2, 1, \text{hasBloodPressure}(?p, ' \text{Prehypertension})) >$
8. $< 2 : \text{hasBloodPressure}(?p, ' \text{Stage1hypertension}) \rightarrow$
 $\text{TELL}(2, 1, \text{hasBloodPressure}(?p, ' \text{Stage1hypertension})) >$
9. $< 2 : \text{hasBloodPressure}(?p, ' \text{Stage2hypertension}) \rightarrow$
 $\text{TELL}(2, 1, \text{hasBloodPressure}(?p, ' \text{Stage2hypertension})) >$
10. $< 2 : \text{hasBloodPressure}(?p, ' \text{Hypotension}) \rightarrow$
 $\text{TELL}(2, 1, \text{hasBloodPressure}(?p, ' \text{Hypotension})) >$

A.3 Diabetes Tester

Initial Facts: $\text{Person}(' \text{Philip}), \text{BloodSugarLevel}('256), \text{hasBloodSugarLevelBeforeMeal}$
 $(' \text{Philip}, '256), \text{greaterThan}('256, '200), \text{lessThanOrEqual}('256, '300)$

Rules:

1. $< 1 : \text{Person}(?p), \text{BloodSugarLevel}(?bsl), \text{hasBloodSugarLevelBeforeMeal}(?p, ?bsl),$
 $\text{lessThan}(?bsl, '130), \text{greaterThan}(?bsl, '80) \rightarrow \text{hasDBCATEGORY}(?p, ' \text{Controlled}) >$

2. $\langle 1 : \text{Person}(?p), \text{BloodSugarLevel}(\text{?bsl}), \text{hasBloodSugarLevelBeforeMeal}(\text{?p}, \text{?bsl}),$
 $\text{lessThanOrEqual}(\text{?bsl}, '200), \text{greaterThan}(\text{?bsl}, '130) \rightarrow$
 $\text{hasDBCcategory}(\text{?p}, ' \text{EstablishedDiabetes}) \rangle$
3. $\langle 1 : \text{Person}(\text{?p}), \text{BloodSugarLevel}(\text{?bsl}), \text{hasBloodSugarLevelBeforeMeal}(\text{?p}, \text{?bsl}),$
 $\text{lessThanOrEqual}(\text{?bsl}, '300), \text{greaterThan}(\text{?bsl}, '200) \rightarrow$
 $\text{hasDBCcategory}(\text{?p}, ' \text{Type2Diabetes}) \rangle$
4. $\langle 1 : \text{Person}(\text{?p}), \text{BloodSugarLevel}(\text{?bsl}), \text{hasBloodSugarLevelBeforeMeal}(\text{?p}, \text{?bsl}),$
 $\text{greaterThan}(\text{?bsl}, '300) \rightarrow \text{hasDBCcategory}(\text{?p}, ' \text{Hyperglycaemia}) \rangle$
5. $\langle 1 : \text{Person}(\text{?p}), \text{BloodSugarLevel}(\text{?bsl}), \text{hasBloodSugarLevelBeforeMeal}(\text{?p}, \text{?bsl}),$
 $\text{lessThanOrEqual}(\text{?bsl}, '60) \rightarrow \text{hasDBCcategory}(\text{?p}, ' \text{Hypoglycaemia}) \rangle$
6. $\langle 2 : \text{hasDBCcategory}(\text{?p}, ' \text{Controlled}) \rightarrow \text{TELL}(3, 1, \text{hasDBCcategory}(\text{?p}, ' \text{Controlled}))$
 \rangle
7. $\langle 2 : \text{hasDBCcategory}(\text{?p}, ' \text{EstablishedDiabetes}) \rightarrow$
 $\text{TELL}(3, 1, \text{hasDBCcategory}(\text{?p}, ' \text{EstablishedDiabetes})) \rangle$
8. $\langle 2 : \text{hasDBCcategory}(\text{?p}, ' \text{Type2Diabetes}) \rightarrow$
 $\text{TELL}(3, 1, \text{hasDBCcategory}(\text{?p}, ' \text{Type2Diabetes})) \rangle$
9. $\langle 2 : \text{hasDBCcategory}(\text{?p}, ' \text{Hyperglycaemia}) \rightarrow$
 $\text{TELL}(3, 1, \text{hasDBCcategory}(\text{?p}, ' \text{Hyperglycaemia})) \rangle$
10. $\langle 2 : \text{hasDBCcategory}(\text{?p}, ' \text{Hypoglycaemia}) \rightarrow$
 $\text{TELL}(3, 1, \text{hasDBCcategory}(\text{?p}, ' \text{Hypoglycaemia})) \rangle$

A.4 Body Temperature

Initial Facts: $\text{Person}(' \text{Philip}), \text{BodyTemperature}(' 104), \text{hasBodyTemperature}(' \text{Philip}, ' 104),$
 $\text{greaterThanOrEqual}(' 104, ' 103)$

Rules:

1. $\langle 1 : \text{Person}(?p), \text{BodyTemperature}(?temp), \text{hasBodyTemperature}(?p, ?temp), \text{lessThan}(?temp, '95) \rightarrow \text{hasFever}(?p, 'Hypothermia) \rangle$
2. $\langle 1 : \text{Person}(?p), \text{BodyTemperature}(?temp), \text{hasBodyTemperature}(?p, ?temp), \text{greaterThan}(?temp, '95), \text{lessThan}(?temp, '99) \rightarrow \text{hasFever}(?p, 'Normal) \rangle$
3. $\langle 1 : \text{Person}(?p), \text{BodyTemperature}(?temp), \text{hasBodyTemperature}(?p, ?temp), \text{greaterThanOrEqual}(?temp, '99), \text{lessThan}(?temp, '101) \rightarrow \text{hasFever}(?p, 'High) \rangle$
4. $\langle 1 : \text{Person}(?p), \text{BodyTemperature}(?temp), \text{hasBodyTemperature}(?p, ?temp), \text{greaterThanOrEqual}(?temp, '101), \text{lessThan}(?temp, '103) \rightarrow \text{hasFever}(?p, 'Hyperthermia) \rangle$
5. $\langle 1 : \text{Person}(?p), \text{BodyTemperature}(?temp), \text{hasBodyTemperature}(?p, ?temp), \text{greaterThanOrEqual}(?temp, '103) \rightarrow \text{hasFever}(?p, 'Hyperpyrexia) \rangle$
6. $\langle 2 : \text{hasFever}(?p, 'Hypothermia) \rightarrow \text{TELL}(4, 1, \text{hasFever}(?p, 'Hypothermia)) \rangle$
7. $\langle 2 : \text{hasFever}(?p, 'Normal) \rightarrow \text{TELL}(4, 1, \text{hasFever}(?p, 'Normal)) \rangle$
8. $\langle 2 : \text{hasFever}(?p, 'High) \rightarrow \text{TELL}(4, 1, \text{hasFever}(?p, 'High)) \rangle$
9. $\langle 2 : \text{hasFever}(?p, 'Hyperthermia) \rightarrow \text{TELL}(4, 1, \text{hasFever}(?p, 'Hyperthermia)) \rangle$
10. $\langle 2 : \text{hasFever}(?p, 'Hyperpyrexia) \rightarrow \text{TELL}(4, 1, \text{hasFever}(?p, 'Hyperpyrexia)) \rangle$

A.5 Pulse Monitor

Initial Facts: $\text{Person}('Philip), \text{Pulse}('120), \text{hasPulse}('Philip, '120), \text{greaterThan}('120, '100)$

Rules:

1. $\langle 1 : \text{Person}(?p), \text{Pulse}(?pulse), \text{hasPulse}(?p, ?pulse), \text{lessThan}(?pulse, '100), \text{greaterThan}(?pulse, '60) \rightarrow \text{hasPulseRate}(?p, 'Normal) \rangle$

2. $\langle 1 : \text{Person}(\text{?p}), \text{Pulse}(\text{?pulse}), \text{hasPulse}(\text{?p}, \text{?pulse}), \text{lessThan}(\text{?pulse}, '60) \rightarrow \text{hasPulseRate}(\text{?p}, 'Abnormal) \rangle$
3. $\langle 1 : \text{Person}(\text{?p}), \text{Pulse}(\text{?pulse}), \text{hasPulse}(\text{?p}, \text{?pulse}), \text{greaterThan}(\text{?pulse}, '110) \rightarrow \text{hasPulseRate}(\text{?p}, 'Abnormal) \rangle$
4. $\langle 2 : \text{hasPulseRate}(\text{?p}, 'Normal) \rightarrow \text{TELL}(5, 1, \text{hasPulseRate}(\text{?p}, 'Normal)) \rangle$
5. $\langle 2 : \text{hasPulseRate}(\text{?p}, 'Abnormal) \rightarrow \text{TELL}(5, 1, \text{hasPulseRate}(\text{?p}, 'Abnormal)) \rangle$

A.6 Ambulance Agent

Initial Facts: Ambulance('KajangTownVan3)

Rules:

1. $\langle 2 : \text{TELL}(10, 6, \text{hasAmbulanceCallFor}(\text{?p}, \text{?loc})) \rightarrow \text{hasAmbulanceCallFor}(\text{?p}, \text{?loc}) \rangle$
2. $\langle 1 : \text{Ambulance}(\text{?amb}), \text{hasAmbulanceCallFor}(\text{?p}, \text{?loc}) \rightarrow \text{isRescuedBy}(\text{?p}, \text{?amb}) \rangle$

A.7 Emergency Monitoring Agent

Initial Facts:

Rules:

1. $\langle 4 : \text{TELL}(1, 7, \text{hasSituation}(\text{?p}, 'Emergency)) \rightarrow \text{hasSituation}(\text{?p}, 'Emergency) \rangle$
2. $\langle 3 : \text{TELL}(9, 7, \text{hasGPSLocation}(\text{?p}, \text{?loc})) \rightarrow \text{hasGPSLocation}(\text{?p}, \text{?loc}) \rangle$
3. $\langle 2 : \text{hasSituation}(\text{?p}, 'Emergency), \text{hasGPSLocation}(\text{?p}, \text{?loc}) \rightarrow \text{hasWarningSign}(\text{?p}, \text{?loc}) \rangle$
4. $\langle 1 : \text{hasWarningSign}(\text{?p}, \text{?loc}) \rightarrow \text{TELL}(7, 10, \text{hasWarningSign}(\text{?p}, \text{?loc})) \rangle$

A.8 OnCall Agent

Initial Facts:

Rules:

1. $\langle 6 : TELL(1, 8, hasSituation(?p, 'OnCall)) \rightarrow hasSituation(?p, 'OnCall) \rangle$
2. $\langle 5 : TELL(9, 8, hasGPSLocation(?p, ?loc)) \rightarrow hasGPSLocation(?p, ?loc) \rangle$
3. $\langle 4 : hasSituation(?p, 'OnCall), hasGPSLocation(?p, ?loc) \rightarrow hasAlarmFor(?p, ?loc) \rangle$
4. $\langle 3 : hasAlarmFor(?p, ?loc) \rightarrow TELL(8, 11, hasAlarmFor(?p, ?loc)) \rangle$
5. $\langle 1 : TELL(13, 8, BurglarAlarm('Beep)) \rightarrow BurglarAlarm('Beep) \rangle$
6. $\langle 1 : TELL(14, 8, BurglarAlarm('Beep)) \rightarrow BurglarAlarm('Beep) \rangle$
7. $\langle 1 : TELL(15, 8, BurglarAlarm('Beep)) \rightarrow BurglarAlarm('Beep) \rangle$
8. $\langle 1 : TELL(16, 8, BurglarAlarm('Beep)) \rightarrow BurglarAlarm('Beep) \rangle$
9. $\langle 2 : BurglarAlarm('Beep) \rightarrow TELL(8, 17, BurglarAlarm('Beep)) \rangle$

A.9 GPS Sensor

Initial Facts: Person('Philip), GPS('KFCKajangTown)

Rules:

1. $\langle 3 : Person(?p), GPS(?loc) \rightarrow hasGPSLocation(?p, ?loc) \rangle$
2. $\langle 2 : hasGPSLocation(?p, ?loc) \rightarrow TELL(9, 7, hasGPSLocation(?p, ?loc)) \rangle$
3. $\langle 1 : hasGPSLocation(?p, ?loc) \rightarrow TELL(9, 8, hasGPSLocation(?p, ?loc)) \rangle$

A.10 Telephone Agent

Initial Facts:

Rules:

1. $\langle 3 : TELL(7, 10, hasWarningSign(?p, ?loc)) \rightarrow hasWarningSign(?p, ?loc) \rangle$
2. $\langle 2 : hasWarningSign(?p, ?loc) \rightarrow hasAmbulanceCallFor(?p, ?loc) \rangle$
3. $\langle 1 : hasAmbulanceCallFor(?p, ?loc) \rightarrow TELL(10, 6, hasAmbulanceCallFor(?p, ?loc)) \rangle$

A.11 Caregiver Agent

Initial Facts: Caregiver('Alice), hasCaregiver('Philip, 'Alice)

Rules:

1. $\langle 2 : TELL(8, 11, hasAlarmFor(?p, ?loc)) \rightarrow hasAlarmFor(?p, ?loc) \rangle$
2. $\langle 1 : Caregiver(?c), hasCaregiver(?p, ?c), hasAlarmFor(?p, ?loc) \rightarrow logAlarm(?c, ?p) \rangle$

A.12 Image Sensor

Initial Facts: Person('Alice), AuthorizedPersonID('1001), hasAuthorizedPersonID('Alice, '1001), UnauthorizedID('2001), hasUnauthorizedID('Bob, '2001)

Rules:

1. $\langle 2 : Person(?p), AuthorizedPersonID(?apid), hasAuthorizedPersonID(?p, ?apid) \rightarrow isAuthorizedPerson(?p, 'Yes) \rangle$
2. $\langle 4 : Person(?p), UnauthorizedID(?suspectID), hasUnauthorizedID(?p, ?suspectid) \rightarrow isAuthorizedPerson(?p, 'No) \rangle$

3. $\langle 1 : isAuthorizedPerson(?p, 'Yes) \rightarrow TELL(12, 13, isAuthorizedPerson(?p, 'Yes)) \rangle$
 \rangle
4. $\langle 3 : isAuthorizedPerson(?p, 'No) \rightarrow TELL(12, 13, isAuthorizedPerson(?p, 'No)) \rangle$
 \rangle

A.13 Motion Detector

Initial Facts: MotionDetector('Yes)

Rules:

1. $\langle 5 : TELL(12, 13, isAuthorizedPerson(?p, 'Yes)) \rightarrow isAuthorizedPerson(?p, 'Yes) \rangle$
2. $\langle 5 : TELL(12, 13, isAuthorizedPerson(?p, 'No)) \rightarrow isAuthorizedPerson(?p, 'No) \rangle$
3. $\langle 1 : MotionDetector(?md), isAuthorizedPerson(?p, 'Yes) \rightarrow BurglarAlarm('NoBeep) \rangle$
4. $\langle 4 : MotionDetector(?md), isAuthorizedPerson(?p, 'No) \rightarrow BurglarAlarm('Beep) \rangle$
5. $\langle 3 : BurglarAlarm('Beep) \rightarrow TELL(13, 8, BurglarAlarm('Beep)) \rangle$
6. $\langle 2 : MotionDetector('Yes) \rightarrow TELL(13, 19, MotionDetector('Yes)) \rangle$

A.14 Gas Detector

Initial Facts: GasDetector('GasLeakage), isGasDetected('GasLeakage, 'Yes)

Rules:

1. $\langle 3 : GasDetector(?gas), isGasDetected(?gas, 'Yes) \rightarrow BurglarAlarm('Beep) \rangle$

2. $\langle 1 : \text{GasDetector}(?gas), \text{isGasDetected}(?gas, 'No) \rightarrow \text{BurglarAlarm}('NoBeep) \rangle$
3. $\langle 2 : \text{BurglarAlarm}('Beep) \rightarrow \text{TELL}(14, 8, \text{BurglarAlarm}('Beep)) \rangle$

A.15 Glass Break Sensor

Initial Facts: $\text{GlassDetector}('GlassBroke), \text{isGlassBroken}('GlassBroke, 'Yes)$

Rules:

1. $\langle 3 : \text{GlassDetector}(?g), \text{isGlassBroken}(?g, 'Yes) \rightarrow \text{BurglarAlarm}('Beep) \rangle$
2. $\langle 1 : \text{GlassDetector}(?g), \text{isGlassBroken}(?g, 'No) \rightarrow \text{BurglarAlarm}('NoBeep) \rangle$
3. $\langle 2 : \text{BurglarAlarm}('Beep) \rightarrow \text{TELL}(15, 8, \text{BurglarAlarm}('Beep)) \rangle$

A.16 Smoke Sensor

Initial Facts: $\text{Smoke}('Smoke), \text{isSmokeDetected}('Smoke, 'Yes)$

Rules:

1. $\langle 3 : \text{Smoke}(?s), \text{isSmokeDetected}(?s, 'Yes) \rightarrow \text{BurglarAlarm}('Beep) \rangle$
2. $\langle 1 : \text{Smoke}(?s), \text{isSmokeDetected}(?s, 'No) \rightarrow \text{BurglarAlarm}('NoBeep) \rangle$
3. $\langle 2 : \text{BurglarAlarm}('Beep) \rightarrow \text{TELL}(16, 8, \text{BurglarAlarm}('Beep)) \rangle$

A.17 Relative Sensor

Initial Facts: $\text{Relative}('Maria), \text{Patient}('Philip)$

Rules:

1. $\langle 2 : \text{TELL}(8, 17, \text{BurglarAlarm}('Beep)) \rightarrow \text{BurglarAlarm}('Beep) \rangle$
2. $\langle 1 : \text{Relative}(?r), \text{Patient}(?p), \text{BurglarAlarm}('Beep) \rightarrow \text{hasRelative}(?p, ?r) \rangle$

A.18 Light Sensor

Initial Facts:

Rules:

1. $\langle 4 : TELL(19, 18, hasOccupancy(?p, 'Yes)) \rightarrow hasOccupancy(?p, 'Yes) \rangle$
2. $\langle 2 : TELL(19, 18, hasOccupancy(?p, 'No)) \rightarrow hasOccupancy(?p, 'No) \rangle$
3. $\langle 3 : hasOccupancy(?p, 'Yes) \rightarrow hasLightFor(?p, 'On) \rangle$
4. $\langle 1 : hasOccupancy(?p, 'No) \rightarrow hasLightFor(?p, 'Off) \rangle$

A.19 Occupancy sensor

Initial Facts: Person('Philip)

Rules:

1. $\langle 5 : TELL(13, 19, MotionDetector('Yes)) \rightarrow MotionDetector('Yes) \rangle$
2. $\langle 4 : Person(?p), MotionDetector('Yes) \rightarrow hasOccupancy(?p, 'Yes) \rangle$
3. $\langle 2 : Person(?p), MotionDetector('No) \rightarrow hasOccupancy(?p, 'No) \rangle$
4. $\langle 3 : hasOccupancy(?p, 'Yes) \rightarrow TELL(19, 20, hasOccupancy(?p, 'Yes)) \rangle$
5. $\langle 3 : hasOccupancy(?p, 'Yes) \rightarrow TELL(19, 18, hasOccupancy(?p, 'Yes)) \rangle$
6. $\langle 1 : hasOccupancy(?p, 'No) \rightarrow TELL(19, 20, hasOccupancy(?p, 'No)) \rangle$
7. $\langle 1 : hasOccupancy(?p, 'No) \rightarrow TELL(19, 18, hasOccupancy(?p, 'No)) \rangle$

A.20 Aircon Sensor

Initial Facts: Aircon('On)

Rules:

1. $< 3 : TELL(21, 20, hasTemperature(?temp, 'Cool)) \rightarrow$
 $hasTemperature(?temp, 'Cool) >$
2. $< 5 : TELL(21, 20, hasTemperature(?temp, 'Hot)) \rightarrow$
 $hasTemperature(?temp, 'Hot) >$
3. $< 6 : TELL(19, 20, hasOccupancy(?p, 'Yes)) \rightarrow hasOccupancy(?p, 'Yes) >$
4. $< 2 : TELL(19, 20, hasOccupancy(?p, 'No)) \rightarrow hasOccupancy(?p, 'No) >$
5. $< 4 : hasOccupancy(?p, 'Yes), hasTemperature(?temp, 'Hot) \rightarrow$
 $hasAirConFor(?p, 'On) >$
6. $< 1 : hasOccupancy(?p, 'No), hasTemperature(?temp, 'Hot) \rightarrow$
 $hasAirConFor(?p, 'Off) >$
7. $< 1 : hasOccupancy(?p, 'No) \rightarrow hasAirConFor(?p, 'Off) >$

A.21 Temperature Level Sensor

Initial Facts: Temperature('28), greaterThan('28, '25)

Rules:

1. $< 1 : Temperature(?temp), greaterThan(?temp, '18), lessThan(?temp, '25)$
 \rightarrow
 $hasTemperature(?temp, 'Normal) >$
2. $< 5 : Temperature(?temp), greaterThan(?temp, '25) \rightarrow$
 $hasTemperature(?temp, 'Hot) >$
3. $< 4 : Temperature(?temp), lessThan(?temp, '18) \rightarrow$
 $hasTemperature(?temp, 'Cool) >$
4. $< 2 : hasTemperature(?temp, 'Cool) \rightarrow TELL(21, 20,$
 $hasTemperature(?temp, 'Cool)) >$

5. $< 3 : hasTemperature(?temp, 'Hot) \rightarrow TELL(21, 20,$
 $hasTemperature(?temp, 'Hot)) >$

Appendix B

The *Onto-HCR* Translated Rules

Welcome to *Onto-HCR*

List of ontologies

=====

1. Patient Care System
2. Home Care System
3. A simple example
4. Patient Care System (Previous Version)
5. Smart Environment Ontology

Please enter your choice : 5

Extracting Axioms From Ontology

=====

*****TBox axioms*****

PhysiologicalData(?x) -> Context(?x)

BodyTemperature(?x) -> PhysiologicalData(?x)

UnAuthorizedPersonID(?x) -> Context(?x)
Alarm(?x) -> Context(?x)
OccupancySensor(?x) -> SmartHomerSensor(?x)
SystolicBP(?x) -> BloodPressureAgent(?x)
Tell13To8BurglarAlarm(?x) -> Alarm(?x)
EmergencyAlarm(?x) -> Alarm(?x)
Tell8To17BurglarAlarm(?x) -> Alarm(?x)
SmartHomerSensor(?x) -> Context(?x)
Person(?x) -> Context(?x)
ImageSensor(?x) -> SmartHomerSensor(?x)
Tell15To8BurglarAlarm(?x) -> Alarm(?x)
EmergencyMonitoring(?x) -> SmartHomerSensor(?x)
AuthorizedPersonID(?x) -> Context(?x)
Informal(?x) -> AuthorizedPerson(?x)
Light(?x) -> Devices(?x)
Aircon(?x) -> Devices(?x)
Formal(?x) -> AuthorizedPerson(?x)
Tell14To8BurglarAlarm(?x) -> Alarm(?x)
Tell13To19MotionDetector(?x) -> SmartHomerSensor(?x)
OnCall(?x) -> SmartHomerSensor(?x)
Caregiver(?x) -> Informal(?x)
BloodSugarLevel(?x) -> DiabetesTester(?x)
GPS(?x) -> PhsiologicalData(?x)
MotionDetector(?x) -> SmartHomerSensor(?x)
AuthorizedPerson(?x) -> Person(?x)
DiastolicBP(?x) -> BloodPressureAgent(?x)
GasDetector(?x) -> SmartHomerSensor(?x)
GlassBreakSensor(?x) -> SmartHomerSensor(?x)

Devices(?x) -> Context(?x)
PatientID(?x) -> Context(?x)
PulseMonitor(?x) -> PhsiologicalData(?x)
BurglarAlarm(?x) -> Alarm(?x)
PC(?x) -> Devices(?x)
DiabetesTester(?x) -> PhsiologicalData(?x)
Temperature(?x) -> SmartHomerSensor(?x)
Patient(?x) -> Person(?x)
PDA(?x) -> Devices(?x)
Ambulance(?x) -> Context(?x)
BloodPressureAgent(?x) -> PhsiologicalData(?x)
Fan(?x) -> Devices(?x)
Relative(?x) -> Caregiver(?x)
Tell16To8BurglarAlarm(?x) -> Alarm(?x)
Nurse(?x) -> Formal(?x)
Smoke(?x) -> SmartHomerSensor(?x)

*****ABox axioms*****

PatientID('P001)
UnauthorizedPersonID('2001)
GasDetector('GasLeakage)
Nurse('Alice)
Caregiver('Maria)
BodyTemperature('104)
OccupancySensor('yes)
PulseMonitor('120)
Person('Philip)
MotionDetector('Yes)

BloodSugarLevel('256)

Temperature('28)

AuthorizedPersonID('1001)

Smoke('smoke)

Ambulance('KajangTownVan3)

DiastolicBP('88)

GlassBreakSensor('GlassBroke)

GPS('KFCKajang)

Aircon('On)

SystolicBP('134)

hasRelative('Philip, 'Alice)

isGlassBroken('GlassBroke, 'Alice)

hasAuthorizedPersonID('Philip, '1001)

hasPatientID('Philip, 'P001)

hasBloodSugarLevelBeforeMeal('Philip, '256)

hasBodyTemperature('Philip, '104)

isSmokeDetected('smoke, 'Yes)

hasDiastolicBP('Philip, '88)

hasGPSLocation('2001, 'KFCKajang)

hasCaregiver('Philip, 'Maria)

logAlarm('Philip, 'KajangTownVan3)

hasAmbulanceCallFor('Philip, 'KFCKajang)

isGasDetected('GasLeakage, 'KFCKajang)

hasUnauthorizedPersonID('Philip, '2001)

isRescuedBy('Philip, 'KajangTown_Van3)

hasSystolicBP('Philip, '134)

hasPulse('Philip, '120)

*****Horn Clause Rules*****

Tell14To8BurglarAlarm(?Beep) -> BurglarAlarm(?Beep)
 Temperature(?temp),lessThan(?temp, "18") -> hasTemperature(?temp, "Cool")
 OccupancySensor(?x) -> SmartHomerSensor(?x)
 BloodSugarLevel(?bsl),Person(?p),hasBloodSugarLevelBeforeMeal(?p,?bsl),
 greaterThan(?bsl, "80"),lessThan(?bsl, "130") ->
 hasDBCategory(?p, "Controlled")
 MotionDetector(?md),Person(?p) -> hasOccupancy(?p, "No")
 Patient(?p),hasDBCategory(?p, "Hypoglycaemia") -> hasSituation(?p, "Emergency")
 Patient(?p),hasBloodPressure(?p, "Stage1hypertension"),
 hasDBCategory(?p, "EstablishedDiabetes"),hasFever(?p, "High")
 -> hasSituation(?p, "OnCall")
 hasBloodPressure(?p, "Normal") -> Tell2To1hasBloodPressure(?p, "Normal")
 Tell15To8BurglarAlarm(?x) -> Alarm(?x)
 Patient(?p),hasFever(?p, "Hyperthermia") -> hasSituation(?p, "OnCall")
 hasGPSLocation(?p,?loc) -> Tell9To7hasGPSLocation(?p,?loc)
 Patient(?p),hasBloodPressure(?p, "Normal") -> hasNotSituation(?p, "Emergency")
 Alarm(?x) -> Context(?x)
 hasGPSLocation(?p,?loc),hasSituation(?p, "OnCall") -> hasAlarmFor(?p,?loc)
 BurglarAlarm(?Beep) -> Tell8To17BurglarAlarm(?Beep)
 Tell3To1hasDBCategory(?p, "EstablishedDiabetes") ->
 hasDBCategory(?p, "EstablishedDiabetes")
 PulseMonitor(?x) -> PhsiologicalData(?x)
 Tell4To1hasFever(?p, "Hypothermia") -> hasFever(?p, "Hypothermia")
 BodyTemperature(?temp),Person(?p),hasBodyTemperature(?p,?temp),

```

greaterThanOrEqual(?temp, "103" ) -> hasFever(?p, "Hyperpyrexia")

BloodPressureAgent(?x) -> PhsiologicalData(?x)

Tell8To17BurglarAlarm(?x) -> Alarm(?x)

Temperature(?x) -> SmartHomerSensor(?x)

AuthorizedPerson(?x) -> Person(?x)

Patient(?p),hasBloodPressure(?p, "'Stage2hypertension") ->
hasSituation(?p, "Emergency")

hasAmbulanceCallFor(?p,?loc) -> Tell10To6hasAmbulanceCallFor(?p,?loc)

PhsiologicalData(?x) -> Context(?x)

Temperature(?temp),greaterThan(?temp, "18" ),lessThan(?temp, "25" ) ->
hasTemperature(?temp, "Normal")

Tell16To8BurglarAlarm(?Beep) -> BurglarAlarm(?Beep)

Tell19To20hasOccupancy(?p, "No") -> hasOccupancy(?p, "No")

Tell5To1hasPulseRate(?p, "Normal") -> hasPulseRate(?p, "Normal")

Patient(?p),hasBloodPressure(?p, "Prehypertension"),hasDBCATEGORY(?p, "Controlled"),
hasFever(?p, "Normal") -> hasNotSituation(?p, "OnCall")

Patient(?p),hasSituation(?p, "OnCall") -> Tell1To8hasSituation(?p, "OnCall")

SmartHomerSensor(?x) -> Context(?x)

Patient(?p),hasDBCATEGORY(?p, "Type2Diabetes"),hasPulseRate(?p, "Abnormal")
-> hasSituation(?p, "Emergency")

Patient(?p),hasFever(?p, "Hyperpyrexia") -> hasSituation(?p, "Emergency")

hasBloodPressure(?p, "Prehypertension") ->
Tell2To1hasBloodPressure(?p, "Prehypertension")

hasDBCATEGORY(?p, "Type2Diabetes") -> Tell3To1hasDBCATEGORY(?p, "Type2Diabetes")

MotionDetector(?x) -> SmartHomerSensor(?x)

hasGPSLocation(?p,?loc),hasSituation(?p, "Emergency") -> hasWarningSign(?p,?loc)

Tell2To1hasBloodPressure(?p, "Stage1hypertension") ->
hasBloodPressure(?p, "Stage1hypertension")

```

hasDBCATEGORY(?p, "Hyperglycaemia") ->
 Tell3To1hasDBCATEGORY(?p, "Hyperglycaemia")
 Patient(?p),hasPulseRate(?p, "Abnormal") -> hasSituation(?p, "Emergency")
 hasGPSLocation(?p,?loc) -> Tell9To8hasGPSLocation(?p,?loc)
 Tell13To19MotionDetector(?x) -> MotionDetector(?x)
 MotionDetector(?md),isAuthorizedPerson(?p, "No") -> BurglarAlarm(?p)
 BurglarAlarm(?Beep) -> Tell15To8BurglarAlarm(?Beep)
 hasOccupancy(?p, "No") -> hasAirconFor(?p, "Off")
 Patient(?p),hasFever(?p, "Hyperpyrexia"),hasPulseRate(?p, "Abnormal")
 -> hasSituation(?p, "Emergency")
 Patient(?p),hasDBCATEGORY(?p, "EstablishedDiabetes"),hasFever(?p, "Hyperpyrexia")
 -> hasSituation(?p, "Emergency")
 hasBloodPressure(?p, "Stage2hypertension") ->
 Tell2To1hasBloodPressure(?p, "Stage2hypertension")
 AuthorizedPersonID(?apid),Person(?p),hasAuthorizedPersonID(?p,?apid) ->
 isAuthorizedPerson(?p, "Yes")
 OccupancySensor(?os),Person(?p) -> hasOccupancy(?p, "No")
 Patient(?p),hasDBCATEGORY(?p, "Hyperglycaemia") -> hasSituation(?p, "Emergency")
 PC(?x) -> Devices(?x)
 Tell5To1hasPulseRate(?p, "Abnormal") -> hasPulseRate(?p, "Abnormal")
 GlassBreakSensor(?x) -> SmartHomerSensor(?x)
 Smoke(?s),isSmokeDetected(?s, "No") -> BurglarAlarm(?s)
 DiastolicBP(?dbp),Person(?p),SystolicBP(?sbp),hasDiastolicBP(?p,?dbp),
 hasSystolicBP(?p,?sbp),greaterThan(?dbp, "90"),greaterThan(?sbp, "140"),
 lessThan(?dbp, "100"),lessThan(?sbp, "160") -> hasBloodPressure(?p, "Stage1hypertension")
 Tell8To21hasAlarmFor(?p,?loc) -> hasAlarmFor(?p,?loc)
 BodyTemperature(?x) -> PhsiologicalData(?x)
 Tell13To19MotionDetector(?Yes) -> MotionDetector(?Yes)

Patient(?p),hasBloodPressure(?p, "Stage2hypertension"),
 hasDBCategory(?p, "Hyperglycaemia"),hasFever(?p, "Hyperpyrexia"),
 hasPulseRate(?p, "Abnormal") -> hasSituation(?p, "Emergency")
 isAuthorizedPerson(?p, "No") -> Tell12To13isAuthorizedPerson(?p, "No")
 Patient(?p),hasFever(?p, "Normal") -> hasNotSituation(?p, "Emergency")
 Smoke(?s),isSmokeDetected(?s, "Yes") -> BurglarAlarm(?s)
 Patient(?x) -> Person(?x)
 hasDBCategory(?p, "Hypoglycaemia") ->
 Tell3To1hasDBCategory(?p, "Hypoglycaemia")
 Tell8To17BurglarAlarm(?Beep) -> BurglarAlarm(?Beep)
 Patient(?p),hasBloodPressure(?p, "Stage2hypertension"),hasDBCategory(?p, "Type2Diabetes")
 -> hasSituation(?p, "Emergency")
 Patient(?p),hasPulseRate(?p, "Normal") -> hasNotSituation(?p, "Emergency")
 Tell10To6hasAmbulanceCallFor(?p,?loc) -> hasAmbulanceCallFor(?p,?loc)
 hasOccupancy(?p, "No") -> hasLightFor(?p, "Off")
 BodyTemperature(?temp),Person(?p),hasBodyTemperature(?p,?temp),
 greaterThan(?temp, "95"),lessThan(?temp, "99") -> hasFever(?p, "Normal")
 hasOccupancy(?p, "Yes"),hasTemperature(?temp, "Hot") -> hasAirconFor(?p, "On")
 Patient(?p),hasDBCategory(?p, "Type2Diabetes"),hasFever(?p, "Normal")
 -> hasSituation(?p, "OnCall")
 Tell4To1hasFever(?p, "Normal") -> hasFever(?p, "Normal")
 hasBloodPressure(?p, "Hypotension") ->
 Tell2To1hasBloodPressure(?p, "Hypotension")
 hasOccupancy(?p, "No") -> Tell19To18hasOccupancy(?p, "No")
 ImageSensor(?x) -> SmartHomerSensor(?x)
 hasFever(?p, "High") -> Tell4To1hasFever(?p, "High")
 hasWarningSign(?p,?loc) -> hasAmbulanceCallFor(?p,?loc)
 BodyTemperature(?temp),Person(?p),hasBodyTemperature(?p,?temp),

greaterThanOrEqual(?temp, "101"),lessThan(?temp, "103") ->

hasFever(?p, "Hyperthermia")

Patient(?p),hasBloodPressure(?p, "Stage1hypertension") ->

hasSituation(?p, "OnCall")

hasWarningSign(?p,?loc) -> Tell7To10hasWarningSign(?p,?loc)

GasDetector(?gas),isGasDetected(?gas, "Yes") -> BurglarAlarm(?gas)

hasTemperature(?temp, "Cool") -> Tell21To20hasTemperature(?temp, "Cool")

hasFever(?p, "Hyperpyrexia") -> Tell4To1hasFever(?p, "Hyperpyrexia")

GasDetector(?gas),isGasDetected(?gas, "No") -> BurglarAlarm(?gas)

MotionDetector(?md),isAuthorizedPerson(?p, "Yes") -> BurglarAlarm(?p)

Tell12To13isAuthorizedPerson(?p, "No") -> isAuthorizedPerson(?p, "No")

Tell4To1hasFever(?p, "Hyperthermia") -> hasFever(?p, "Hyperthermia")

Tell21To20hasTemperature(?temp, "Cool") -> hasTemperature(?temp, "Cool")

Patient(?p),hasBloodPressure(?p, "Prehypertension"),hasDBCcategory(?p, "Hyperglycaemia"),hasPulseRate(?p, "Abnormal") -> hasSituation(?p, "Emergency")

hasBloodPressure(?p, "Stage1hypertension") ->

Tell2To1hasBloodPressure(?p, "Stage1hypertension")

EmergencyAlarm(?x) -> Alarm(?x)

BurglarAlarm(?Beep) -> Tell13To8BurglarAlarm(?Beep)

AuthorizedPersonID(?x) -> Context(?x)

DiastolicBP(?x) -> BloodPressureAgent(?x)

BloodSugarLevel(?x) -> DiabetesTester(?x)

Patient(?p),hasDBCcategory(?p, "EstablishedDiabetes") ->

hasNotSituation(?p, "Emergency")

PDA(?x) -> Devices(?x)

Patient(?p),hasSituation(?p, "Emergency") -> Tell1To7hasSituation(?p, "Emergency")

Tell3To1hasDBCcategory(?p, "Hypoglycaemia") -> hasDBCcategory(?p, "Hypoglycaemia")

BurglarAlarm(?x) -> Alarm(?x)

Patient(?p),hasBloodPressure(?p, "Prehypertension"),
 hasDBCATEGORY(?p, "EstablishedDiabetes"),hasFever(?p, "Normal") ->
 hasNotSituation(?p, "Emergency")

Tell13To8BurglarAlarm(?Beep) -> BurglarAlarm(?Beep)

Tell2To1hasBloodPressure(?p, "Hypotension") ->
 hasBloodPressure(?p, "Hypotension")

Tell15To8BurglarAlarm(?Beep) -> BurglarAlarm(?Beep)

BloodSugarLevel(?bsl),Person(?p),hasBloodSugarLevelBeforeMeal(?p,?bsl),
 greaterThan(?bsl, "300") -> hasDBCATEGORY(?p, "Hyperglycaemia")

Nurse(?x) -> Formal(?x)

EmergencyMonitoring(?x) -> SmartHomerSensor(?x)

DiastolicBP(?dbp),Person(?p),SystolicBP(?sbp),hasDiastolicBP(?p,?dbp),
 hasSystolicBP(?p,?sbp),greaterThan(?dbp, "80"),greaterThan(?sbp, "120"),
 lessThan(?dbp, "90"),lessThan(?sbp, "140") ->
 hasBloodPressure(?p, "Prehypertension")

Smoke(?x) -> SmartHomerSensor(?x)

Tell19To18hasOccupancy(?p, "Yes") -> hasOccupancy(?p, "Yes")

BurglarAlarm(?Beep) -> Tell14To8BurglarAlarm(?Beep)

hasOccupancy(?p, "Yes") -> hasLightFor(?p, "On")

MotionDetector(?md),Person(?p) -> hasOccupancy(?p, "Yes")

hasPulseRate(?p, "Normal") -> Tell5To1hasPulseRate(?p, "Normal")

Formal(?x) -> AuthorizedPerson(?x)

Tell2To1hasBloodPressure(?p, "Prehypertension") ->
 hasBloodPressure(?p, "Prehypertension")

SystolicBP(?x) -> BloodPressureAgent(?x)

Patient(?p),hasBloodPressure(?p, "Prehypertension") ->
 hasNotSituation(?p, "Emergency")

Tell9To8hasGPSLocation(?p,?loc) -> hasGPSLocation(?p,?loc)

OnCall(?x) -> SmartHomerSensor(?x)
 Patient(?p),hasBloodPressure(?p, "Hypotension") -> hasSituation(?p, "Emergency")
 MotionDetector(?md),OccupancySensor(?os),Person(?p) -> hasOccupancy(?p, "Yes")
 GPS(?loc),Person(?p) -> hasGPSLocation(?p,?loc)
 Ambulance(?x) -> Context(?x)
 Tell1To7hasSituation(?p, "Emergency") -> hasSituation(?p, "Emergency")
 Informal(?x) -> AuthorizedPerson(?x)
 Tell2To1hasBloodPressure(?p, "Normal") -> hasBloodPressure(?p, "Normal")
 PatientID(?pid),Person(?p),hasPatientID(?p,?pid) -> Patient(?p)
 Caregiver(?x) -> Informal(?x)
 Light(?x) -> Devices(?x)
 hasAlarmFor(?p,?loc) -> Tell8To21hasAlarmFor(?p,?loc)
 Tell3To1hasDBCcategory(?p, "Type2Diabetes") -> hasDBCcategory(?p, "Type2Diabetes")
 hasOccupancy(?p, "No") -> Tell19To20hasOccupancy(?p, "No")
 MotionDetector(?Yes) -> Tell13To19MotionDetector(?Yes)
 hasOccupancy(?p, "No"),hasTemperature(?temp, "Hot") -> hasAirconFor(?p, "Off")
 Tell21To20hasTemperature(?temp, "Hot") -> hasTemperature(?temp, "Hot")
 DiastolicBP(?dbp),Person(?p),SystolicBP(?sbp),hasDiastolicBP(?p,?dbp),
 hasSystolicBP(?p,?sbp),greaterThan(?dbp, "60"),greaterThan(?sbp, "90"),lessThan(?dbp,
 "80"),
 lessThan(?sbp, "120") -> hasBloodPressure(?p, "Normal")
 hasOccupancy(?p, "Yes") -> Tell19To18hasOccupancy(?p, "Yes")
 hasPulseRate(?p, "Abnormal") -> Tell5To1hasPulseRate(?p, "Abnormal")
 BloodSugarLevel(?bsl),Person(?p),hasBloodSugarLevelBeforeMeal(?p,?bsl),
 lessThanOrEqual(?bsl, "60") -> hasDBCcategory(?p, "Hypoglycaemia")
 BodyTemperature(?temp),Person(?p),hasBodyTemperature(?p,?temp),
 lessThan(?temp, "95") -> hasFever(?p, "Hypothermia")
 Tell3To1hasDBCcategory(?p, "Hyperglycaemia") ->

hasDBCategory(?p, "Hyperglycaemia")

GlassBreakSensor(?g),isGlassBroken(?g, "No") -> BurglarAlarm(?g)

hasFever(?p, "Hypothermia") -> Tell4To1hasFever(?p, "Hypothermia")

Tell19To20hasOccupancy(?p, "Yes") -> hasOccupancy(?p, "Yes")

Person(?p),PulseMonitor(?pulse),hasPulse(?p,?pulse),greaterThan(?pulse, "110")

-> hasPulseRate(?p, "Abnormal")

GasDetector(?x) -> SmartHomerSensor(?x)

Person(?p),UnauthorizedPersonID(?suspectID),hasUnauthorizedPersonID(?p,?suspectid)

-> isAuthorizedPerson(?p, "No")

hasOccupancy(?p, "Yes") -> Tell19To20hasOccupancy(?p, "Yes")

DiastolicBP(?dbp),Person(?p),SystolicBP(?sbp),hasDiastolicBP(?p,?dbp),

hasSystolicBP(?p,?sbp),lessThan(?dbp, "60"),lessThan(?sbp, "90")

-> hasBloodPressure(?p, "Hypotension")

GlassBreakSensor(?g),isGlassBroken(?g, "Yes") -> BurglarAlarm(?g)

Fan(?x) -> Devices(?x)

Patient(?p),hasDBCategory(?p, "Controlled") -> hasNotSituation(?p, "Emergency")

hasDBCategory(?p, "EstablishedDiabetes") ->

Tell3To1hasDBCategory(?p, "EstablishedDiabetes")

UnauthorizedPersonID(?x) -> Context(?x)

Tell12To13isAuthorizedPerson(?p, "Yes") -> isAuthorizedPerson(?p, "Yes")

BodyTemperature(?temp),Person(?p),hasBodyTemperature(?p,?temp),

greaterThanOrEqual(?temp, "99"),lessThan(?temp, "101") -> hasFever(?p, "High")

BurglarAlarm(?Beep),Patient(?p),Relative(?r) -> hasRelative(?p,?r)

Tell9To7hasGPSLocation(?p,?loc) -> hasGPSLocation(?p,?loc)

DiabetesTester(?x) -> PhysiologicalData(?x)

hasDBCategory(?p, "Controlled") -> Tell3To1hasDBCategory(?p, "Controlled")

Person(?p),PulseMonitor(?pulse),hasPulse(?p,?pulse),lessThan(?pulse, "60")

-> hasPulseRate(?p, "Abnormal")

Tell3To1hasDBCategory(?p, "Controlled") -> hasDBCategory(?p, "Controlled")
 Caregiver(?c),hasAlarmFor(?p,?loc),hasCaregiver(?p,?c) -> logAlarm(?c,?p)
 Patient(?p),hasDBCategory(?p, "Type2Diabetes") -> hasSituation(?p, "OnCall")
 Tell4To1hasFever(?p, "High") -> hasFever(?p, "High")
 DiastolicBP(?dbp),Person(?p),SystolicBP(?sbp),hasDiastolicBP(?p,?dbp),
 hasSystolicBP(?p,?sbp),greaterThan(?dbp, "100"),greaterThan(?sbp, "160")
 -> hasBloodPressure(?p, "Stage2hypertension")
 Temperature(?temp),greaterThan(?temp, "25") -> hasTemperature(?temp, "Hot")
 BurglarAlarm(?Beep) -> Tell16To8BurglarAlarm(?Beep)
 hasFever(?p, "Hyperthermia") -> Tell4To1hasFever(?p, "Hyperthermia")
 Person(?p),PulseMonitor(?pulse),hasPulse(?p,?pulse),greaterThan(?pulse, "60"),
 lessThan(?pulse, "100") -> hasPulseRate(?p, "Normal")
 Tell7To10hasWarningSign(?p,?loc) -> hasWarningSign(?p,?loc)
 Patient(?p),hasFever(?p, "Hypothermia") -> hasSituation(?p, "Emergency")
 BloodSugarLevel(?bsl),Person(?p),hasBloodSugarLevelBeforeMeal(?p,?bsl),
 greaterThan(?bsl, "200"),lessThanOrEqual(?bsl, "300") ->
 hasDBCategory(?p, "Type2Diabetes")
 Tell4To1hasFever(?p, "Hyperpyrexia") -> hasFever(?p, "Hyperpyrexia")
 BloodSugarLevel(?bsl),Person(?p),hasBloodSugarLevelBeforeMeal(?p,?bsl),
 greaterThan(?bsl, "130"),lessThanOrEqual(?bsl, "200") ->
 hasDBCategory(?p, "EstablishedDiabetes")
 Relative(?x) -> Caregiver(?x)
 Ambulance(?amb),hasAmbulanceCallFor(?p,?loc) -> isRescuedBy(?p,?amb)
 Aircon(?x) -> Devices(?x)
 GPS(?x) -> PhsiologicalData(?x)
 Tell1To8hasSituation(?p, "OnCall") -> hasSituation(?p, "OnCall")
 Devices(?x) -> Context(?x)
 Tell19To18hasOccupancy(?p, "No") -> hasOccupancy(?p, "No")

Person(?x) -> Context(?x)

hasFever(?p, "Normal") -> Tell4To1hasFever(?p, "Normal")

Tell2To1hasBloodPressure(?p, "Stage2hypertension") ->
hasBloodPressure(?p, "Stage2hypertension")

isAuthorizedPerson(?p, "Yes") -> Tell12To13isAuthorizedPerson(?p, "Yes")

Bye...