



University of
Nottingham

UK | CHINA | MALAYSIA

A Rule-based Framework for Developing Context-Aware Systems for Smart Spaces

THESIS SUBMITTED TO THE UNIVERSITY OF
NOTTINGHAM
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

*Submitted by
Ijaz uddin*

School of Computer Science
University of Nottingham
Malaysia Campus

June 2019

Abstract

Context-aware computing is a mobile computing paradigm that helps designing and implementing next generation smart applications, where personalized devices interact with users in smart environments. Development of such applications are inherently complex due to these applications adapt to changing contextual information and they often run on resource-bounded devices. Most of the existing context-aware development frameworks are centralized, adopt clientserver architecture, and do not consider resource limitations of context-aware devices. This thesis presents a systematic framework to modelling and implementation of multi-agent context-aware rule-based systems on resource-constrained devices, which includes a lightweight efficient rule engine and a wide range of user preferences to reduce the number of rules while inferring personalized contexts. This shows rules can be reduced in order to optimize the inference engine execution speed, and ultimately to reduce total execution time and execution cost. The use of the proposed framework is illustrated using five different case scenarios considering different smart environment domains.

Acknowledgements

First of all, I thank ALLAH for his blessings upon me and giving me the courage to undertake the PhD course in such a prestigious university. I thank Dr Abdur Rakib for his excellent supervision and trust in me to give me the chance to work for him. Throughout my PhD, he has never shown any lack of interest; instead, every meeting made me more and more energetic to overcome the PhD. It is possible only through his continuous efforts that I can finish my PhD. I also thank Dr Thomas Maul for being the most caring academician. It was only possible due to Dr Thomas that i travelled to Vietnam for conference and won the best paper award. Further, I am especially thankful to Dr Iman Yi Liao, for being my formal supervisor and trusted in me. Moreover, I thank my parents, brother Altaf and my wife for their prayers and patience. I also acknowledge that my daughter Zarghoona Khattak sacrificed her childhood for me and spent time alone and far from her cousins and friends when she needed it the most.

I thank my senior, Dr Mahfuz ul Haque for his excellent directions given to me when I was stuck in study or was feeling down. Dr Mahfuz taught me a lot, and I am thankful for his support during and after his PhD. It was not possible without my lab mates especially, Moataz, Tuong and Koah for their continuous support(laughing when there was a deadline and work to do) and patience(when there was work to do, and they were waiting to go out for eating kabab). I also thank Moataz to work with me on the great for-loop problem we had.

In the end, I must say that I really enjoyed my time in UNMC only due to the best supervisor, best environment and best friends around.

Publications From Thesis

- **An efficient rule-based distributed reasoning framework for resource-bounded systems.** Mobile Networks and Applications. 2018 Aug 18:1-8. (Journal)(IF 3.2)
- **A Resource-Aware Preference Model for Context-Aware Systems** November 2017, ICCASA 2017 (Springer) ,Tam ky, Vietnam
- **Modeling and Reasoning about Preference-Based Context-Aware Agents over Heterogeneous Knowledge Sources** June 2017, Mobile Networks and Applications (Journal)(IF 3.2)(Accepted)
- **A Preference based application framework for resource bounded Context aware agents** June 2017, International conference on mobile and wireless technologies, Kuala Lumpur, Malaysia
- **Modelling and Reasoning About Context-Aware Agents over Heterogeneous Knowledge Sources** (contributor) (Best Paper Award) April 2017, Context aware systems and application (Springer), Vietnam
- **A Framework for Implementing Formally Verified Resource-Bounded Smart Space Systems** Dec 2016, Mobile Networks and Applications (Journal)(IF 3.2)
- **Resource-Bounded Context-Aware Applications: A Survey and Early Experiment** March 2016, Nature of computation and communication (Springer), Vietnam

Contents

1	Introduction	1
1.1	Motivation	4
1.2	Research Questions	6
1.3	Problem Statement	8
1.4	Aims and Objectives	10
1.5	Methodology	11
1.6	Major Contributions	12
1.6.1	A Customized Algorithm Development For Efficient Running On Small Devices	13
1.6.2	Preferences	13
	Context-Based Preference	14
	Derived Preference	14
	Live Preferences	15
	Preferences On Rules	15
1.6.3	Rules Generation	15
	Desktop Based Rule Generator	15
	Rule From Ontology	16
1.6.4	Practical Implementation Of The Framework	16
1.7	Thesis Outline	16
2	Context And Context-Awareness	18
2.1	Introduction	18
2.2	Context	19
2.3	Context-Aware Systems	21
2.3.1	Context-Aware Multi-Agent Systems	22
2.3.2	Types Of Context-Aware Computing	24
	Location-Aware Computing	24
	User-Aware Computing	25

	Energy-Aware Computing	26
	Resource-Aware Computing	26
	Environment-Aware Computing	27
	Situation-Aware Computing	28
2.4	Application Of Context-Aware Systems	29
2.5	General Design Of Context-Aware Application	34
2.6	Architectural Style Of Context-Aware Systems	35
2.7	Context Modelling Approaches	38
2.8	Ontologies	40
	2.8.1 Representation Of Ontologies In Computer Science	42
	2.8.2 Ontology Design Approaches	43
	2.8.3 Ontology Designs And Languages	44
2.9	Rule-Based System	48
	2.9.1 Components Of RBS	50
	2.9.2 Rule Execution	52
	2.9.3 Conflict Resolution	53
2.10	Discussion	54
3	Related Work	56
3.1	Introduction	56
3.2	Context-Aware Development Platform	57
	3.2.1 Origin Model	57
	3.2.2 ContextJ	60
	3.2.3 JCAF	61
	3.2.4 OPEN	62
3.3	Rule-Based Systems And Existing Rule Engines	64
	3.3.1 Prolog	64
	3.3.2 Jess	65
	3.3.3 Clips	66
3.4	Rule-Based Systems For Mobile Devices	68
3.5	RETE Algorithm	73
	3.5.1 Analysis Of RETE Algorithm	75
	3.5.2 The Match Problem	79
3.6	Discussion	81
4	Proposed Platform, Context Acquisition Scheme And Algorithm	83
4.1	Introduction	83

4.2	Proposed Platform	84
4.3	Context Acquisition Model	85
4.3.1	Context Acquisition	86
4.3.2	Context Acquisition Frequency	86
4.3.3	Context Acquisition Methods From Source	88
4.3.4	Context Reasoning	88
4.4	Model Of Context-Aware Systems	89
4.4.1	Rule Format	91
4.4.2	Context-Aware Systems as Resource-Bounded Agents	92
4.5	Algorithm Design	95
4.5.1	Match: Conflict Set Generation	96
4.5.2	Select: Conflict Resolution	100
4.5.3	Act: Execution Of The Selected Rule Instance	100
4.5.4	Working Memory Updating	103
4.5.5	Communication And Subroutine Handling	104
4.5.6	Time And Space Complexity Of Core Algorithms	108
	Match Algorithm Complexity	108
	Conflict Resolution Algorithm Complexity	111
	Execution Of Rule Complexity	112
4.6	Rule Generation And System Design	114
4.6.1	Web Based Rule Generator	114
4.6.2	Onto-HCR Protégé Plugin For Heterogeneous Ontologies Development Environment	117 118
	Working Mechanism Of The Plugin	119
4.7	Discussion	120
5	Preferences	125
5.1	Introduction	125
5.2	Preference In Context-Aware Agents	128
5.2.1	Context Set	129
5.2.2	Context Monitor	130
5.2.3	Preference Set Generator	131
5.2.4	Working Mechanism	131
5.3	Type Of Preferences	132
5.3.1	Context-Based	132
5.3.2	Derived Context-Based Preference	133
5.3.3	Live Preference	133
5.3.4	Preference Set Generation	135

5.4	Case Studies	136
5.4.1	Case Study 1	136
	Facts, CoI and Rules transition	141
	Discussion	141
5.4.2	Case Study 2	143
5.4.3	Case Study 3	148
	Context-based Preferences	148
	Rule-based Preferences	149
5.4.4	Case Study 4	150
	System Setup	151
	Ontologies and Agents	152
	Scenarios	154
	Agents Details	154
	Scenario Execution Smart Patient, Home And Office	157
	Dry Run Analysis	158
5.4.5	Case Study 5	161
5.4.6	Case Studies Results	163
5.5	Discussion	164
6	Conclusion And Future Work	173
6.1	Conclusion	173
6.2	Future Work	177
6.2.1	Hardware Advancement	177
6.2.2	Unified Ruling System	178
6.2.3	Reversal Of Preference Set	178
6.2.4	Working Memory Limitation And Updation	179
	Distinct WM	179
	Average of the preference sets	179
	User assigned	180
A	Complete List Of Rules	181

List of Figures

2.1	Nurse call system architecture	32
2.2	WorldMate flight details	33
2.3	Centralized architecture	36
2.4	Distributed architecture	37
2.5	Stand alone architecture	38
2.6	Rule-base schematic view	51
3.1	RETE Network Illustration	76
4.1	Distributed problem solving	93
4.2	Step by step rule and fact processing	99
4.3	Different matching scenarios of the proposed algorithm	99
4.4	Rule-base initialization	115
4.5	Validation of Horn-clause rules	116
4.6	Facts interface with auto suggestion	116
4.7	Distributed semantic knowledge translation process	117
4.8	Enabling the plugin in protégé	121
4.9	Selecting ontology for translation	122
4.10	Editing the translated file	123
5.1	Preference generation overview	129
5.2	Derived context based preference set generation	134
5.3	Live preference change detection	135
5.4	Smart office ontology	138
5.5	Patient care ontology	139
5.6	Ontologies with corresponding agents	152
5.7	Patient Care Ontology	153
5.8	Preferences enabled	162

5.9 Preferences disabled 162

List of Tables

2.1	Abstract layer of context-aware system	34
3.1	Comparison Table For Rule Engines. (Y: Yes, N: NO, NA: Not Applicable)	72
3.2	Algorithm Left Activation Running Cost	78
3.3	Algorithm Right Activation Running Cost	78
4.1	Summary of proposed system	87
4.2	One possible run of the system	95
4.3	Agent ID table	108
4.4	Conflict set generation Algorithm Complexity	109
4.5	Algorithm Conflict Resolution Running Cost	111
4.6	Cost for Executing a Selected Rule Instance	113
5.1	Sample rules from two ontologies with identifiers	140
5.2	Preference set transition	141
5.3	Preference set transition with rules	142
5.4	Blood pressure, heart rate rules and example rules	147
5.5	Some example rules of Agent 1	150
5.6	Preference set transition	151
5.7	Complete Agent Wise Rules and Reductions in Rules	160
5.8	Execution of Smart Patient (A)	166
5.9	Execution of Smart Patient (B)	167
5.10	Execution of Smart Patient (C)	168
5.11	Execution of Smart Home (A)	169
5.12	Execution of Smart Home (B)	170
5.13	Execution of Smart Home (C)	171
5.14	Execution of Smart Office	172

A.1	Set of rules for Smart Patient	181
A.1	Set of rules for Smart Patient	182
A.1	Set of rules for Smart Patient	183
A.1	Set of rules for Smart Patient	184
A.1	Set of rules for Smart Patient	185
A.1	Set of rules for Smart Patient	186
A.1	Set of rules for Smart Patient	187
A.1	Set of rules for Smart Patient	188
A.2	Set of rules for Smart Home	188
A.2	Set of rules for Smart Home	189
A.2	Set of rules for Smart Home	190
A.3	Set of rules for Smart Office	191
A.3	Set of rules for Smart Office	192
A.3	Set of rules for Smart Office	193
A.4	Set of rules for Smart Shopping Cart	193
A.5	Set of rules for Smart Shopping Cart	194

Chapter 1

Introduction

The last few decades have seen exponential growth and change in a variety of computer-related technologies. Computers have evolved from big bulky mechanical machines into lightweight lightning fast laptops and tablet computers. Tasks that were once possible only on big computers can be done easily on small desktop computers or even handheld devices such as PDA or smartphone. While computers were successfully prospering, there was the beginning of mobile phones. In 1973 Motorola first introduced handheld telephone device [1]. It was not until 1980 that the mobile use was slowly transferring to public use. Later on, mobile phones with small size and longer battery life were introduced. Such mobile phones were easy to carry and were available at the market at affordable prices for public purchases. The technology took it further and the late 20th century has witnessed the transfer of mobile phone into a smartphone. Smartphones are capable of carrying out

our daily routine tasks, which were earlier possible on computers or other similar devices only, such as browsing the Internet, social networking, taking photos or making videos etc [2]. With the advancements of the smartphone combined with feature-rich software, applications and Internet connectivity make it easier for people to share their experiences using social networking applications, including VoIP services, free messaging and call applications, to name some [3].

With the availability of such devices, it seems that Weiser's vision about the 21st-century computer was right, with providing the idea of ubiquitous computing [4], where different computers are connected and communicate with each other, anywhere any time. Ubiquitous computing accommodated smartphones very well. The smartphones being small devices with adequate computing power further gives the ubiquitous computing mobility. Moreover, a variety of communication modes available on smartphones makes it more connected with other devices. A typical smartphone can use, call, SMS, Wi-Fi, blue-tooth, Infrared, hotspot and more communication modes, which makes the users able to employ different services whenever and wherever needed. Along with various high-tech features, a smartphone is also equipped with a wide range of sensors, including a global positioning system (GPS), shake sensors, accelerometers, and proximity sensors [5]. These sensors that accommodate a user in his/her daily life can further be used in a large variety of applications, which can provide user related and surrounding information. This information is formally called as *contexts* when

used within the scope of context-aware computing paradigm. These sensors can be integrated in a way to provide enough user information, including the user's location, time, movement, and surrounding environmental information. When equipped with a suitable communication mechanism, it can also enhance interaction between the user, application and other devices [6]. Since a smartphone is quite an independent device itself and analogous to agent-based computing, where a device works independently, a smartphone can easily qualify as an agent with the help of some code or application. The smartphones or other devices that are used to implement such applications may act as *intelligent agents* for a particular scenario of an application. Thus smartphones and agent-based technology can provide tremendous benefits for the development of more intelligent systems or in other words, a better context-aware system. With the concept of ubiquitous computing and context-aware capabilities, it is more suited to devise a system that can be deployed anywhere, have context-aware skills and can infer output or new context based on its current information. To achieve such a mechanism, there are quite different ways to give a machine the ability to think and then decide, on behalf of another person or device. One such approach is by using a rule-based system, which makes use of certain rules to drive the system. Having such an intelligent system on a small device can significantly improve the people's life standard (especially disabled people), with very little user intervention or efforts required.

This research work aims to study different context-aware systems, rea-

soning mechanism, matching algorithms and rule-based systems and their implementation on small devices. Further to propose a qualified mechanism for providing a variety of services on resource-limited devices. The rest of this chapter presents the motivation for developing a context-aware system framework which adopts the rule-based reasoning mechanism with a significant focus on implementation on resource-bounded devices, e.g., smartphones. Research questions, problem statements, methodology and major contributions are also highlighted. At the end of this chapter, the thesis structure is outlined.

1.1 Motivation

Ubiquitous computing is a way that can be defined as “whenever, wherever” computing. What happens behind providing such services can take different approaches. Ubiquitous computing is a natural bind with the context-aware system. Since in a ubiquitous system, different devices are connected and can provide service anytime anywhere. The context-aware terminology further automates this anytime anywhere mechanism. Imagine a user enters his/her office, and all the devices are tuned into his/her settings automatically, for example, the lights are turned on, the fan and air-con automatically adjust the temperature according to the weather outside. The comfort level of the seat is adjusted, and the TODO list is brought into the user’s attention. Such personalization can be achieved when ubiquitous computing

embraces context-aware computing. Since all of these settings are taking place on behalf of the user, and attaining such mechanism, it is necessary to make the system intelligent enough so that it can consider all the available context and based on these contexts take some decision on behalf of the user. Such intelligence can be achieved if the knowledge of the user is encoded into the framework, and for that reason, rule-based reasoning is used. Rules can be used to encode an expert knowledge in the simplest form. However, the combination seems perfect but not easy. As a rule-based system consumes not only much memory but also needs a powerful computer for processing. While in today's world of mobility, a smartphone is not a match for the high-tech desktop system. Some approaches tackled the issue by using a cloud computing for processing, which leaves the smartphone merely a sending and receiving device. This motivates the research in the direction where the work is almost negligible. Further, the literature also suggests a lack of mature work in this direction. Therefore, with the facility of personalization the expert-knowledge and reducing the rules redundancy and developing a resource-friendly framework, will provide new trends in this research. Resource-bounded devices may include devices which have limited memory, processing power, operating power, battery life, bandwidth, communication channels, are handheld such as a smartphone or smart chips running Android. In this thesis, such devices are also referred to as tiny devices. In the same fashion, tiny memory relates to devices with limited memory with no possibility of expansion. The motivational factors

that contributed towards taking this research also provided us with research questions, discussed in the next section.

1.2 Research Questions

This thesis intends to find out the relation of the rule-based system with context-awareness. Specifically, how to use rule-based systems to develop context-aware applications. Categorically, we split our perspective into different objectives and incrementally address them one by one. First, to study the rule-based systems itself and then to find out what role it can play in terms of functionally correct context-aware applications development. Primarily, what challenges to expect or can be faced when the resources of the host machine are minimal. Resources can be the memory, computation power, battery life to name some, but not limited to them. Secondly, how to define an algorithm which is efficient and light enough to be used on resource-bounded devices and must have the reasoning power to process the rules of knowledge base provided and derive accurate output. Since the smart spaces are composed of multiple devices or agents, and each agent plays its role to overcome some problem. For this purpose, the agents reason, derive, deduce, and share context within the limitation of its resources. An efficient algorithm alone is not enough to overcome the resource constraints that we expect to encounter and improve performance. Therefore, to improve the performance, we need some mechanism to reduce the redundancy issues, that is common

in rule-based systems. Reducing redundancy within rules will directly affect the overall running time of an agent. Instead of going through all the rules, it will only focus on a small set of rules with higher chances of getting fired. To approach these difficulties, we need to develop some understanding of rule-based systems and compare them with the conventional approaches. As what benefits it brings along with itself, so its usage can be favoured, especially in the resource-bounded environment. We need to study, how the rule-based system supports the context-aware paradigm, and what are some current literature that focuses on reasoning facilities with resource management point of view. Further, it is essential to develop some techniques which can directly impact the system performance. Since rules are the main driving factor of any rule-based system, and the number of rules available in the knowledge base can take a sufficiently long time to search for a particular rule instance to fire. We tried to work on this area also, and did some work to reduce the redundancy while keeping the integrity and correctness of output accurate.

After addressing such issues, we will be able to provide an efficient rule-based algorithm for the development of context-aware systems in a resource-constrained environment. This thesis has both theoretical and practical contributions, the theory is discussed with the help of different concepts, methods and design framework, while a working prototype of various components is also developed as part of the implementation.

1.3 Problem Statement

In today's era of technology and the complex nature of application development for a broader range of domains, it is quite challenging to come up with a concrete solution for context-aware systems. Context-aware systems itself are quite complex, as the context cannot be predicted in advance nor the output of the system. Further, it becomes more complicated when the solution is sought to be on different agents or in a multi-agent system in a distributed fashion. In a multi-agent environment, the agents work together to accomplish a single task or multiple tasks. A certain level of difficulty is increased when the agents are capable of moving from place to place. As each agent is considered as a small device with minimal resources, the scenario becomes more complicated when these agents not only process context but also acquire the context, process the context, reason about the context and provide some new context and share the context with other agents. In order to reason, an agent should possess a rule engine sufficiently enough to produce accurate results while consuming fewer resources. Although some efforts are being made in different approaches to our knowledge, however, there is no any viable and practical solution for the development of context-aware applications based on the rule-based systems which are specially tailored for tiny devices with limited resources and working in distributed multi agent-based setting. Using rule-based system brings its complexity such as matching problem, redundancy of rules, higher run time complexity to name some.

Therefore, it is vital to address these issues if the framework is intended for devices which have limitations on the resources. Also, when context-aware systems are used, they are highly favourable to give personalized services to the user. Accommodating personalization in rules need preferences based on user choice. Usually, the development for such frameworks takes place on a high-end system with an abundance of resources. Therefore, the resource-bounded area has been ignored widely. Having such a framework in hand can cater to the needs of consumers residing in remote locations, warzones, disaster-hit areas to name some. Depending on the nature of the area, the framework can be loaded with the rule-base to help users with minimal resources. Some work has been initiated in [7, 8] but the work seems in a primitive stage, or there is no advancement recently in their work. The authors of [7] argue that there are no pure mobile based rule-engines, and the authors of [8] further states that there does not exist any comprehensive design and development tool which covers all the aspects of context-aware applications in the mobile platform including, e.g., methodology, language, inference engine and communication protocols. They further state that such development environment is essential and will benefit both the researchers and developers. As recent as 2018, the study of rule engines in [9] state that there are very few rule engines that might work on a mobile platform, which further motivates us to take upon the research problem. Further, it states the porting option for different rule-engines, although the literature does not recommend the porting. Most importantly, they speak in terms

of rule engine without resource constraints. It is pertinent to mention that, rule-engines porting for Android is an active research work, but the resource limitation constraints are not added whenever the literature has discussed the portability of rule-engines.

This thesis aims to propose not only the framework but practically implement and demonstrate a complex case scenario involving different agents and rules. The framework uses rule-based reasoning agents to reason about contextual information and to infer new contexts. The framework is mainly, but there is also a manual rule writing mechanism. The core of the framework depends on the proposed algorithm, which has very few lines of codes and has relatively low time and space complexity. To further reduce the load from the rule-engine, a novel approach of preferences on the rules is proposed, which reduces the number of rules for processing without affecting the integrity of the output. Focus is given on the design of the system being in a distributed way and can move from place to place.

1.4 Aims and Objectives

This research work aims to provide a formally correct framework to implement resource-bounded systems for smart spaces. This aim will be achieved by focusing on the objectives on resource-bounded devices, especially Android devices. Smart spaces integrate many technologies, and as discussed, the rule-based system will provide the reasoning capability while the context-

aware computing will perceive the environment and bring the related context to be processed. In order to have a formally verified framework, the core of the framework will be driven by the custom developed algorithm, which will use meagre resources and memory in terms of running and space complexity. Moreover, in order to further lower the computation cost, a novel approach for preferences will be introduced. Preferences will give two benefits; first, it will reduce the size of the rules base so that the number of processable rules or the inputs will be minimum. Secondly, it will personalize the services for the user. The preferences will be available in different modes to mimic human-like behaviour and to provide more personalized services to another human.

1.5 Methodology

An outline of the concrete methodology is given below:

1. We use both ontology-based approach and manual approach to represent contexts and rule-based reasoning to infer implicit context from a provided set of explicit contexts. The system is modelled as a context-aware, multi-agent system. Agents act as reasoning agents, and they reason over a knowledge-based using First order Horn-Clause rules.
2. We develop an algorithm, which is specifically tailored for the need of resource-bounded devices. The algorithm has comparatively low

running time and space complexity.

3. We enhance the rule-engine by introducing three levels of preferences, i.e. Context-based, Derived context based and live context based. Each has its own and different application method.
4. We enhanced the OWL API based Onto-HCR tool to accommodate multiple ontologies and translate them into first-order Horn-clause rules format.
5. We developed and integrated the D-Onto-HCR tool into a plugin which can be used in the Protégé to translate any ontology/ies and edit them before using them.
6. We developed three case scenarios which are validated both logically and practically to check the integrity of the framework. Case studies are different and include dry-run analysis and implementation results.

1.6 Major Contributions

The major contributions of this thesis are provided, which ultimately work together to provide efficient context-aware rule-based solution on devices with limited or low resources.

1.6.1 A Customized Algorithm Development For Efficient Running On Small Devices

One of the main contributions of this research work is the development of the matching algorithm. Our algorithm is designed to save space and run efficiently on devices with little computation power and memory. Not only does it consume low resources, but the algorithm itself also has a quite lower number of lines in practice. This algorithm is the main driving force behind the whole framework, and it can run on different agents seamlessly. Theoretically, any Java-enabled device can run the algorithm as a special precaution were taken while development in order to only use the very basic Java in order to run it on as much device as possible. Being developed for Android Jelly Beans means that it can practically run on 98 per cent of Android devices. While designing this matching algorithm for the rule, we also found that the performance is directly proportioned to the input rules. None of the previous work, even for big systems, has tried to minimize the rules which are redundant for a given set of context, although some work has proposed idea to select rules based on the contents of working memory. We have proposed a novel approach to reduce redundancy using preferences.

1.6.2 Preferences

Preference modelling is our novel approach in order to make the size of the rule-base as low as possible. Given a large size of rules can slow down the

whole system. The literature, as we will discuss in a later chapter, suggests different techniques such as rule ordering to avoid overhead. We tackle this issue with different techniques of preferences. In either case, preference completely, remove the rule which has no chance of being fired, hence saving the vital resources. The main idea of user preference is to select a subset of rules based on preferences. By doing this the inference engine, instead of going through all the rules, will only process selected rules. Some introduction to different kind of preferences is provided below.

Context-Based Preference

In preference based on the context, a subset is selected before the system startup which gathers the rules based on the user preferences provided. The system excludes the rules that the user does not want for a particular set of contexts. However, as the system moves on new context are derived, and there is a possibility that the newly derived contexts can match a rule which has been excluded. Therefore we enhanced the preference mechanism and accommodated the derived preferences.

Derived Preference

These are the preferences that a user expect that it might come in the future. , and once we found that a context has been derived, and the system recognizes it. A new set of rules is generated, and the system can move on without losing the integrity of results, which are based on derived context.

Live Preferences

Live preferences are applied when some context is continuous, and the value can change, e.g. GPS data. A GPS sensor can send us different coordinates every time it sends a signal when a user is moving. This enables a user to select different rules based on his/her present location.

Preferences On Rules

The uniqueness of the preference is also traced into the rule-based design. Whenever in a conflict set, we encounter two or more rules, we execute a rule which has a higher priority. As the user give preference to one rule over another rule, the rule with a high preference score will be fired.

1.6.3 Rules Generation

We tried to make the system as simple as possible. The rules can be either written by a desktop-based interface or straight from ontology.

Desktop Based Rule Generator

The desktop-based, web-enabled software enables a designer to develop a whole rule-base faster. It is used for a manual rule-base development when the rules are not available from an ontology, or the ontology does not exist at all. The user must be familiar with the system in order to make the rules accordingly. The software does detect syntax errors for designer comfort.

Desktop-based rule generator is normally used for a small set of rules.

Rule From Ontology

Rules can be extracted from ontologies, which are supported by our framework. We have developed a Protegé plugin, which allows translation of an OWL 2 RL ontology augmented with SWRL rules into a set of plain text Horn-clause rules. These translated rules with minor user annotation can be used to design the agents. The toolkit supports inputting multiple ontologies and allows heterogeneous sources of rules that can be translated into a single output plain text Horn-clause rules.

1.6.4 Practical Implementation Of The Framework

We have practically developed the whole system and implemented it for the validation. For that reason, different devices are attached to the framework. Some devices are real sensors, while some are simulated sensors. The dry run of the system elaborates every step involved.

1.7 Thesis Outline

In this thesis, Chapter 2 and 3 mainly provide a detailed literature review by first introducing the important concepts and then related work. Chapter 2 covers a detailed view of the notion of context and the terminologies which are based on the term context such as context-awareness and context-aware

system, while the latter part of the chapter details the rule-based system. Context-aware agents are introduced, followed by some areas where the application of a context-aware system is implemented. Chapter 2 further discuss different design styles for context-aware applications and context modelling approaches, which include a brief introduction to ontologies. Later part of the chapter familiarizes the concept of a rule-based system, its different components and stages of execution. Based on the understanding from chapter 2, chapter 3 provides a related work. In chapter 3, different platforms related to the context-aware system, rule-based system or combination of both on desktop systems or Android systems are discussed. RETE algorithm is discussed in detail, along with problems associated with the usage of the RETE algorithm. Chapter 4 describe the proposed platform, the methods by which a context is received, followed by the proposed algorithms in order to make our context-aware system based on the RBS concept. Since chapter 4 details the base of thesis contribution, extended framework and contributions are discussed in Chapter 5 illustrated with very detailed case scenarios based on different agents and the response of the system in a different context. The last chapter concludes the thesis and provides some future directions for the research.

Chapter 2

Context And Context-Awareness

2.1 Introduction

In this chapter, the notion of context and terms which are based on the context, e.g. context-aware are explained. Various terms related to or based on the context along with the architectural style are elaborated. Towards the end of the chapter, related work context modelling approaches are addressed with emphasis on the ontological approach.

2.2 Context

The word *context* is not a new word or a true computer-related word. In general, its linguistic meaning according to the Oxford Dictionary¹ is “*The circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood.*” While in computer science it has a bit different meaning. The definition of context is evolving, as the context definition itself needs a domain, scenario, background or some setting upon which a context can be defined. In this thesis, the context is considered within the premises of context-aware systems. According to the Schilit and Theimer[10], who also first term the concept of context-aware systems regard the context like location, identities of people nearby, objects and changes made on these objects. Schilit further refined the definition [11] and also added the important aspects of context such as where you are, who you are with, and what resources are nearby. Still, there can be some missing components when we derive the context concerning user activity. As user activity is also an important aspect of a context. A more elaborate and simple definition is tossed by [12] and considered the context as a subset of physical and conceptual states of interest to a particular entity. Further definitions of context emerged with time and maturity of the context-aware systems. The widely accepted and a generalized definition is given by Dey et al. [13] as “*Context is any information that can be used to characterize the situa-*

¹<https://en.oxforddictionaries.com/definition/context>

tion 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.” However, this definition is also regarded as very general, and it is difficult to apply in operational meaning. Winograd [14] stresses that “something is context because of the way it is used in interpretation, not due to its inherent properties.” Context is, in fact, a bit tricky term to define, e.g., what is a context and what is not a context. Something that may be considered as a context in one setting may not be considered the context in different settings [15]. More definitions are provided where it is understood that the definition is depended on the domain it is used. Some cross domain definitions are tossed by different research work such as [16, 17] sensor networks [18, 19], retrieving information systems [20], nomadic computing [21] and vastly in artificial intelligence [22, 18, 12]. To elaborate more, context can be further specified by its entities [23]. For example, for a person context, its entities can be the person’s identity, location and his/her surroundings. Similarly, his/her Environmental context can be a day, night, noise, raining, sunshine, cloudy and weather of the area to name some. Physical context can be area, time, date, and location. When these different contexts are made to affect the system output, it contributes towards the context-aware systems.

2.3 Context-Aware Systems

The term *context-aware computing* was first introduced by Schilit and Theimer [10]. They define the context-aware system as

“The ability of a mobile user’s applications to discover and react to changes in the environment they are situated in”

The problem with their definition is being very specific according to Dey [24]; therefor Dey defined the context-aware [25] as

“A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task”

The context-aware computing paradigm emerged in the early 1990s with the introduction of small mobile devices. In 1992, Olivetti Lab’s active badges used the infrared badger assigned to staff members for tracing their locations in the office, and according to the locations calls were forwarded [26]. Further developments lead to the development of various context-aware frameworks such as Georgia Tech’s Context Toolkit [27]. In recent years, more research has been carried out, and advanced context-aware systems exist [28], and the contributions to research and development over the years promise a bright future of such systems. Generally, context-aware systems may interact with humans and/or other devices. Further, they often exhibit complex adaptive behaviours; they are highly decentralized and can naturally be implemented

as multi-agent systems. An agent is a piece of software that requires to be reactive, pro-active, and that is capable of autonomous action in its environment to meet its design objectives [29]. Non-human agents in such a system may be running a straightforward program. However, they are increasingly designed to exhibit flexible, adaptable and intelligent behaviour. Agents can accumulate both human and non-human devices. While in the context of this thesis, an agent is considered a device which may or may not have the inference capacity. It can refer to a simple sensor or a device which can infer new context. The combination of such agents in a setting to accomplish a task makes a context-aware multi-agent system.

2.3.1 Context-Aware Multi-Agent Systems

As context depends on the domain, it represents. While designing/developing context-aware systems, there are various methods that a platform can adopt as a middleware [30] e.g., agent-based techniques. Especially in a distributed environment, in context-aware application development, the key role from the development side is that of agent programming. Agents are defined in the artificial intelligence field [31] as

“An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors”.

Another definition of the agents [32] is also the same in meaning but a more elaborate one.

“An agent is a persistent software entity dedicated to a specific purpose. ‘Persistent’ distinguishes agents from subroutines; agents have their ideas about how to accomplish tasks, their agendas. ‘Special purpose’ distinguishes them from entire multifunction applications; agents are typically much smaller.”

An agent itself is a piece of software that processes contexts, and agents may receive data from a sensor or from another agent which contains the information regarding a particular context.

According to the functionalities of agents provided in [33], an agent is supposed to be autonomous, i.e., it controls its execution all by itself without the intervention of any external entity. It should be social and should interact with other agents. They are reactive to the changes in the environment, and lastly, they are supposed to be pro-active and should have goal-directed behaviour.

Agents send/receive data from other sensors or agents, which is known as low-level-context or raw data as it is merely understandable. This raw data is then transformed into high-level context by combining them or generating meaning from it [34]. For example, GPS provides raw data in a numerical format for any geographical location. This is a low-level context, the application that transforms it into a high-level data, knows the logic of interpreting the value on the map and pinpoint the location in a meaningful format such

as street name or a park name which is a high-level context to make it understandable for the user. In our case, agents can also communicate via message passing considering high-level contexts.

Categorically there are many types of context-aware systems, which can be combined for more elaborate system development. Some of the categories are defined here while this is not a complete list of available possibilities. Instead, it gives the reader a broader idea of how vast the possibilities can reach.

2.3.2 Types Of Context-Aware Computing

Context-awareness has different types. Also, various types of contexts (low-level) can be combined to gather more precise context (high-level) about a subject. Some of the commonly used context types are discussed below.

Location-Aware Computing

As the name indicates, it is the form of context-aware computing, which makes use of a user's current location or the location of the sensor/agent installed. According to EDUCASE [35] it refers to systems that can sense the current location of a user or a device and change its behaviour based on his/her location. For instance, in a hospital, it can change the device volume to silent when it senses the location of the doctor as "in the patient room". Recent work by [36] focuses on context-aware authorization using the location of the user in a building. The authors make use of the smartphones

to authorize a user based on his/her location in the building. The authors claim to provide a scenario that converts a physical space into a smart space.

User-Aware Computing

User-aware computing refers to context-aware computing, which considers user preferences. The system should be aware of the various entities associated with the user, e.g., mood, health, status (includes walking, driving or running). In order to understand the user aware computing, consider an example from [37] where authors explain it with the difference between two instances: first when a user is walking to his/her office alone in the morning, the user aware system response should be one. In the second scenario, when the same user is walking on the same road with his/her toddler, in the evening, the system should respond the other way. User-aware computing can be combined with the location-aware context to provide more precise results.

In the earlier example of location-aware computing, which finds a doctor location as in the patient room. Combining the user-aware system will further define if the doctor is attending a patient, giving a prescription or have just visited the empty room. Taking these into consideration, more precise services can be offered.

Energy-Aware Computing

Energy-aware computing can be explained with a straightforward example from the daily life of turning a cell phone into energy saver mode when the power level falls below a certain threshold value. Energy-aware computing is a broad research topic for cloud computing and data, as observed in [38, 39] and a survey [40]. These research works present various energy aware techniques in the platforms mentioned above from sensors, communication, and resource allocation perspective for better energy management. It should use some decision mechanism to turn off devices in order to consume less energy but maintaining the availability of data should be a priority. It focuses on saving energy and keeps the energy cost at a minimum for data centres. Moreover, the energy-aware system should be aware of the power consumption of a device, or a combination of devices and should be intelligent enough to save as much energy as possible without affecting the services.

Resource-Aware Computing

In resource-aware computing, a system should be well informed of its resources. The resources may be abundant, or there may not be any bound on usage. A system can monitor, continuously and on-demand, its resources and use them accordingly [41]. The authors explain an example of resource aware computing with a task to scan a database of X-ray images stored on a remote server and to get those X-rays which shows a lung cancer. It can be achieved by downloading all the X-rays and then process them locally for

diagnosis. This might be wasting the resources of a network by transferring all the images. Another way to achieve the same task is to remotely isolate the X-rays on the server by sending the selection algorithm to the server and only download the images which are relevant for the diagnosis. This method will reduce the network requirements. The authors further explain that a more flexible way is to send the selection procedure to the database, which, after detecting the relevant X-rays return only the file sizes of the X-rays that have to be downloaded. After that, considering the bandwidth and network resources, the main program may choose to download the images or the program itself may shift to the main server for further processing in order to save the network resources. The authors also agree that processing on the main server might slow down the server behaviour.

Environment-Aware Computing

The environment in computing is a broad term, and it can be used in various contexts. As proposed in research by Microsoft [42] a projector that can sense its environment, makes a 3D model of its surrounding and then can find various gestures of users, out of its created environment. These gestures are further converted for various operations. Environment-aware computing can sense the environment of a user to find if it is day time or sleeping time at night and switch profiles of the user smartphone accordingly. Similarly, the noises surrounded by the user are also accounted for in the user's environment, which can be used to adjust the user's cell phone volume. The

environment can also be used as in [43] which adjust the mobile device various properties by sensing the environment first. These properties may include security settings, filters and status of instant messaging.

In the hospital scenario, much better service can be provided when a smart-phone first detects its environment as a hospital and then the location-aware computing provides a specific location within the hospital. When user contexts are added, it can further narrow down the output toward better context-aware computing.

Situation-Aware Computing

It provides the context information that is related to the situation of a subject. Situation-Aware computing in health care can be a system which monitors the patient's situation in critical condition and acts accordingly [44]. The authors in [45] argue that context-aware system should be well aware of the user situation. A system that is based on location awareness may not identify the situation of the user. A user can be tracked by his/her location to be in his/her living room. However, being in the living room is not enough context unless other context types support it. The context-aware response may differ depending on the situation in the living room as if a user is with family or colleagues in the living room can derive different outputs, based on his/her current situation. The situation can also be based on the machines as the authors in [46] presents a framework for Big Data (huge data sets that

may require computations to find patterns.) to respond timely towards the evolving situations.

2.4 Application Of Context-Aware Systems

Almost two decades ago, the concept of context-aware application started with the advancement in the field. Context-aware application [47] is defined as

“Applications that change their behaviour according to the user’s present context—their location, who they are with, what the time of day is, and so on”

Context-aware applications receive the contexts from the sensors attached to the device, and then according to various techniques, it may react or deduce some output. The smartphone with a variety of embedded sensors becomes quite a reasonable device to run context-aware applications. Initially, smartphones were equipped with a camera. Over time GPS sensors were installed, which can also provide information regarding the location of an image taken or to provide position related information (Geotagging) [48]. With time, proximity, shake sensors, distance calculators from images and various other sensors were added. These sensors made the smartphone an ideal platform for context-aware applications development. From the developer’s perspective, to qualify an application as a good context-aware application, there are some guidelines [49], such as:

- Relevant context information for the application.
- To make the application intelligent enough to understand the obtained information from context with respect to its environment.
- To make the application behaviour change with respect to the new information from the context according to the environment.

There are also context-aware applications that keep track of user's preferences and recommends applications that the user might prefer. On a typical smartphone we can install such context-aware applications [50, 51, 52]. These recommend or launches applications to the user according to his/her preferences. As in [53], the authors provided a technique to recommend applications to the user using the data mining and data analysis technique Bayesian. It proposes a model which instead of the basic preferences use the situation adaptive application category recommending system and uses variables such as stance, location, time and date. It also keeps track of applications that are most used, time of the day when an application is activated. These all accumulate to a system which recommends applications to its user according to his/her preferences. Apart from the recommendation applications, the following examples show how context-aware applications can help humans in different scenarios.

- **Intelligent Patient Care Application**

This example combines various types of aware-technologies in order to present a more precise and accurate functional health care system. The example

presented in [54] has five different steps. Step one shows when a nurse comes to a patient bed, and there is an indication on the floor known as “active zone. A nurse has to stand in the active zone in order to exchange the context. Step two defines that the bed itself is also a context-aware bed where a display is available that is used by the patient for entertainment purpose while the nurse and doctors can use the same display for monitoring patient’s health progress. It also knows about the patient and the person standing near the bed, e.g., doctor and nurse. Step three involves the context-aware pill container, which itself is a communication and context-aware device and recognises its patient. Step four defines the prescription mechanism, i.e., dosage for patient and also logs the attending nurse under whose supervision the prescription is taken. While step five shows how a nurse logs out by just moving away from the patient bed’s active zone.

- **An Ambient-Intelligent Nurse Call System**

This application is the result of a research work which is used for providing care to the patients in a hospital environment. It mainly gathers raw data or low-level context from various sensors that are installed on the patient side. When the system has adequate data available, it can further monitor a patient’s situation, and in case of the patient’s discomfort, it seeks the caregiver or nurse’s attention. It tries to find out the best caregiver available to the patient. A smartphone application is developed for caregivers through which they are accessible and can receive their calls for any patient call. The caregivers respond using the application, and the patient is attended in a

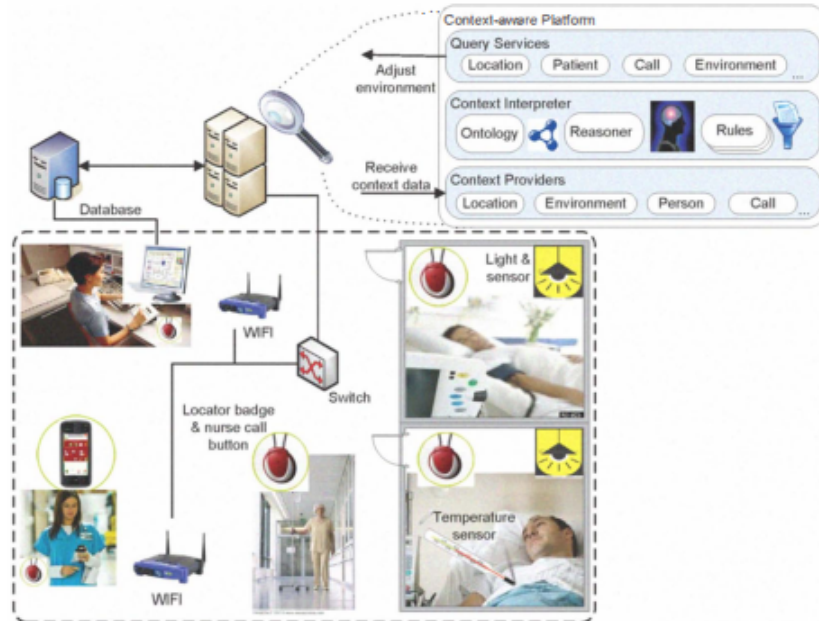


Figure 2.1: Nurse Call system architecture [55]

quicker way [55]. The graphical representation of the health care system is provided in Figure 2.1

- **World Mate**

According to the website, more than ten million users rely on WorldMate² application for their trips arrangement. All a user has to do is to send the confirmation email received for flight to the application. WorldMate transforms these emails into a neat and clean, readable format. It shows the data in the format as provided in Figure 2.2. Also, it alerts in real time should there be any cancellation or delay of the flight and suggest alternatives to

²World Mate www.worldmate.com

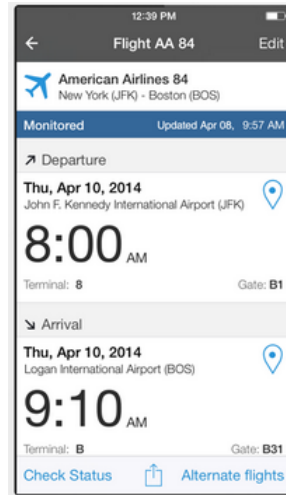


Figure 2.2: WorldMate flight details

the user. It also keeps track of the user's history, and according to the previous knowledge, it provides hotels information when on a trip. Further, the application is also integrated with a professional social networking website, LinkedIn³. It automatically finds contacts from his/her social network in nearby areas and informs the user about his/her contact details. These applications and the frameworks, work on different setups. As commercial applications, they have servers and well-established setup to cater to the user. There are different ways the context-aware application can be designed in terms of its internal structure.

³LinkedIn www.linkedin.com

Application
Storage
Preprocessing
Raw data retrieval
Sensors

Table 2.1: Abstract layer of context-aware system

2.5 General Design Of Context-Aware Application

The context-aware system makes use of various sensors. The low-level context is acquired from sensors and then passed for further processing towards the server or module. The received data is processed and stored according to requirements of the application. There are various architectures defined for the context-aware systems, but the basic structure remains almost the same. A typical abstract layered architecture [56] is shown in Table 2.1. The application layer provides the output to the user as a result of a change in instances or event-driven changes. It is the interface for the user to interact with the system. Storage provides sensors information stored to be retrieved by the clients of the system or by third-party software. Pre-processing interprets the information, validate the reasoning, extract data and provide relevant information to the storage. For example, a GPS coordinates received from sensors are converted into location name. Raw data layer exchanges data with sensors; it has query functions and also holds the results of the queries.

It uses specific API or protocols to request data from the sensor layer. Sensors are the main nodes which retrieve the context from the surrounding environment, e.g., temperature, pressure, noise, light, location and like.

2.6 Architectural Style Of Context-Aware Systems

In terms of context-aware systems, the approach of development depends upon the requirement as in every software. Some application may work well in a distributed environment while some may perform better in a centralised environment. There are some guidelines which should be kept in mind when choosing between centralised, distributed or standalone approach.

- Centralized approach

In this method, a central server is responsible for handling all the requests, storage, processing, and providing outputs. The data is stored mainly on a single context database which has adequate resources for operations. Sensors and devices are attached directly to the context server. If two devices need to communicate with each other, the communication is done via simple protocol as there is no complex networking involved. In this setup, the quality of the architecture depends on the context server [57]. The powerful server will ensure better communication and processing. The centralised system has the benefits of cost-effectiveness and easy management in term of software

and hardware. However, the whole system crashes in case the server fails. Also, congestion can cause to shut down the system in case of overloaded server behaviour by clients [58]. A typical example of a centralised approach is presented in Figure 2.3.

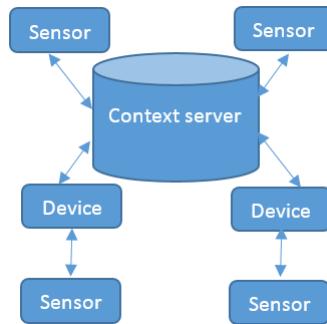


Figure 2.3: Centralized architecture

- Distributed architecture

In distributed context-aware system, context data is stored in more than one place. Each device has its context information and is connected like a mesh. Devices can communicate with each other by the ad-hoc method of communication [58]. Availability of multiple context devices makes it easy to bypass any device that has failed in communication or creating congestion in the network. Although the distributed approach may be a little expensive to implement as every node has its resources, which may include processor and memory. However, unlike the centralised system, if one device fails, the whole system runs fine [57] which increases its robustness. It is easy to isolate

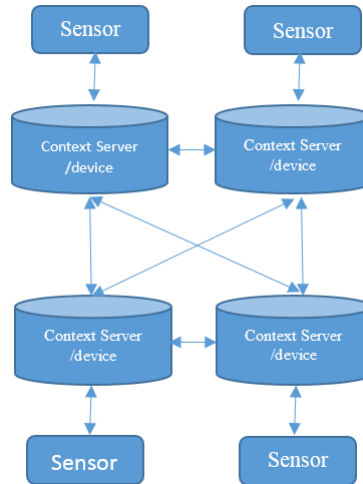


Figure 2.4: Distributed architecture

the problematic node and repair it without disturbing the overall service. A graphical illustration is provided in Figure 2.4.

- Standalone approach

The standalone approach is ideal for a small organisation, which requires limited context details. It is composed of a single device/ sensor and storage. It does not share its information with any other device. It has a limited working capacity and is designed for specific requirements or domain-specific applications [57, 58]. The graphical presentation of the standalone system is provided in Figure 2.5.

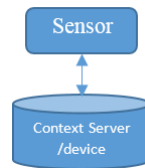


Figure 2.5: Stand alone architecture

2.7 Context Modelling Approaches

System development is a process by which software engineers transform the logic into computer understandable format. In system development, a software development team is deployed to make software for an organisation or some business. The software team then go through some primitive approaches toward development in which they first analyse the system they want to develop. Get the business flows, business rules, how each step is taken in the manual system along with other details if any. After that, the designer designs the system, which is a blueprint for a programmer. The role of a programmer is then to code the designed flow, and at the end, a team of testers have to test the working prototype before deploying [59]. While in context-aware system development, unfortunately, there is no such formal methods or mature platform available and a programmer has to deal with the low-level programming of sensors and actuators [60]. Sufficient knowledge of artificial intelligence is expected from the programmer, which may be based on the logical structures of a provided scenario. The context-aware system can be developed by various programming paradigms, but the important point in the context-aware system is the context model, that shows how

context data can be stored and defined in a machine processable form [61]. Some of the common practices for context modelling are

- **Key-value:** it is simple to implement and is quite flexible and easy to manage when the application size is small. It is not recommended for the development of scalable applications or complex applications. The emphasis in this model is mainly toward the application rather than the context data. The structure is also not well organised.
- **Markup Scheme:** Markup scheme is flexible, structure without any standard for structuring and the validation is possible using schema. Context modelled using this technique can be transferred via a network.
- **Object Based:** Using programming techniques for objects, context can be modelled, although the design principles govern the standards for programming context model. Object-based programming allows the programmer to represent the context in programming code level. The context can be manipulated in run time and is stored in the memory.
- **Logic-based:** Logic-based context modelling allows a programmer to implement logical reasoning techniques. It is also used to generate the high-level context from the low-level context. New facts can be added to working memory based on the current facts of the working memory, or the new facts that arrive from external sources, e.g., sensors. Logic-based programming can be based on a series of rules that might execute when a data is received from sensors. These rules may trigger

further rules for execution. In general context-aware systems do not involve a user for basic input/output operations; instead, it senses the environment using sensors and based on these contexts continue to produce output.

- **Ontology based:** Ontology-based context modelling can expressively represent the context. It is application independent and can be shared with other technologies. Though the representation of context can be a bit complex. It can be used to define domain knowledge, and the structure is also based on the context relationships as defined in the ontology. The data can be stored separately from the structure of the ontology. This can also be integrated with logical rules, which helps to design context-aware systems using rule-based programming.

Ontologies being a subject of interest in the following chapters is discussed with detail in the next section.

2.8 Ontologies

An ontology is an explicit, formal specification of shared conceptualization [62]. Where explicit means that every concept must be defined while formal states that it should be machine understandable and consensus should be developed according to context. Another definition of ontology is defined in the research work [63] as an ontology is a model of a domain that introduces vocabulary relevant to the domain and uses this vocabulary to specify the

relationships among them.

For example, if two persons are talking about Jaguar in a car showroom, it means they are discussing the car brand of Jaguar. Similarly, if two persons are talking about Jaguar in a zoo or jungle, most of the chances are that they are talking about the animal Jaguar. So in order to understand the Jaguar in its context, the communication has to develop a shared consensus about which Jaguar is the communication being held.

From the artificial intelligence perspective, according to Tom Gruber, an AI specialist at Stanford University as of writing this article [64], defines ontology as an explicit specification of a conceptualization. In the same article, the author describes that when declarative formalism is used to define the knowledge of a domain, then the set of objects created forms a universe of discourse which can carry classes, functions and relations among others. The objects and their relationships among them are reflected in its representational vocabulary, with which knowledge-based programs represent the knowledge.

2.8.1 Representation Of Ontologies In Computer Science

In computer science, ontologies are defined using classes, relations and instances. Classes define the concepts which can be related to other concepts. Classes are further described using attributes and name-value pairs. The interrelation of classes can be defined as a class *Person* has two different subclasses such as *Student* and *Professor*. *Professor* gives *Lecture* and *Student* has to attend the *Lecture* means that both are connected to the *Lecture* but in totally different way. Relations are special attributes whose values are objects of other classes. For relations and attributes, constraints define the allowed values, while constraints itself are rules. For example a *Person* has a subclass of *Male* and *Female* but a *Person* cannot opt both value at the same time for a single person. So a rule like $Male \wedge Female$ will give result of empty set.

Instances of a class define the various individuals of an ontology. For example a *Person* (class) having a profession as *Professor*(subclass) has to take a *Course*(subclass) of a *Seminar*(class) so its timing are *Thursday* (instance) at *11 AM* (instance) at *Room no 3* (Instance). Individuals are the basic components of an ontology. It gives ontology concrete objects such as animals or people and likewise.

2.8.2 Ontology Design Approaches

In the article [65], authors provide a guideline in ontology development:

- (i) Identifying classes for the ontology.
- (ii) Taxonomic ordering of the classes should be followed.
- (iii) Defining properties.
- (iv) Lastly insertion of values.

The authors' further state that there is no agreed-upon method for ontology development and the process is iterative. There are various ways of defining the ontology design. In [66], the authors created an ontology for their smart space. The authors described two methods of ontology building that are the bottom-top and top-bottom approaches, which are discussed further.

- Bottom-up approach

In this approach, the ontologies are first created for the smaller objects, after that the high-level abstracts classes are used to define the desired ontology development. In this approach, the particular classes are created first, and then based on these classes, similar classes are grouped as more general concepts, and gradually it develops into a whole ontology. For example, in hospital system it will first define the *Nurse* class, then a *Physician* then *Specialist* along the hierarchy, and group them as a generalized concept of *Caregiver*.

- Top-bottom approach

It is the opposite of the Bottom-Up approach, and it defines the superclasses first and then comes toward more specific subclasses. The above example can be implemented in reverse form as the superclass will be first defined as a caregiver then specialist, general physician, nurse down to the bottom in the hierarchy.

2.8.3 Ontology Designs And Languages

- OWL

OWL is the description logic-based ontology language. The authors of the research work [67] discussed the web ontology language (OWL) from the semantic web perspective and stated that the OWL is semantic markup language for ontologies that provides a formal syntax and semantics for them. The W3C provides different standards for OWL, mainly OWL 1 and OWL 2. OWL 1 has further three flavours as OWL Lite, OWL DL and OWL Full. The OWL Lite is not much expressive when compared to its other two flavours while OWL DL is more expressive and it is based on the descriptive logic. OWL full is a more expressive form of the OWL. The Authors further describe OWL 2 as a more efficient and tractable reasoning standardization. Within the OWL 2 DL which is based on the descriptive language, there are further three more, namely OWL 2 EL, OWL 2 QL and OWL 2 RL. The authors use the OWL 2 RL and SWRL languages for defining Rules (see 5.5.1)

and ontologies in their work. Adding to these, the book [68] also define the three variations of OWL, i.e. Lite, DL and Full. The book defines the OWL Full as a superset of OWL DL which removes some of the restriction that can be found in the OWL DL but it introduces some computational traceability. In practice, OWL Lite is quite sufficient for programming ontologies.

- RuleML

The official mission statement of the RuleML ⁴ states that

”To develop RuleML as the canonical language system for Web rules through schema-defined syntax, formal semantics, and efficient implementations.

The overarching, modular schema specification of the RuleML system includes the Deliberation RuleML and Reaction RuleML families. RuleML is thus used to exchange knowledge bases and queries across rule-based systems, map between Web ontologies, and inter-operate across dynamic network behaviours of workflows, services, and agents”.

In the research work [69] the design rationale of RuleML is discussed with great details. In the International Conference on Artificial Intelligence PRICAI 2000, RuleML initiative took place by the leading experts of the field. The RuleML initiative aims to develop an open XML/RDF based rule lan-

⁴RuleML Wiki
http://wiki.ruleml.org/index.php/RuleML_Home#RuleML_as_a_Bridge

guage for the exchange of rules between various platform, e.g. software components and web databases. Rule plays a significant part in the AI, logic programming besides other fields. The role of the rule has a great place in the semantic web, where data is not only presented but is tried for automation and reuse. Rules markup for the semantic web has a great research interest as rules are identified as one of its design issues. RuleML was initiated on the already existing markup languages, and it has further inspired other projects that are also based on the rule markups. The RuleML community has shown interest in the development of standardized rule language which might have its translators along with other tools.

- SWRL

To achieve more reasoning capacity from OWL, SWRL is used. SWRL is a Semantic Web Rule Language combining OWL and RuleML. SWRL is an expressive rules language, and it enables writing rules using OWL-DL concepts. It is semantically based on description logic. Moreover, to improve its expressive power, SWRL can also be used to extend OWL2RL with first order user-defined rules. SWRL structure can be written, as shown below:

$$\text{Rule Body (Antecedent)} \rightarrow \text{Rule Head (Consequence)}$$

SWRL accommodates number of positive conjunctions in antecedent and consequence part, however disjunctions are not allowed. There exist the problem of undecidability in SWRL as the rules are made on the first-order horn clauses rules. To resolve the undecidability issue, the concept of safe

rules is used, which only allow the rules which are known concepts.

A simple code snippet of SWRL is provided below.

```
<ruleml:imp>
<ruleml:_rlab ruleml:href="#example"/>
<ruleml:_body>
<swrlx:individualPropertyAtom swrlx:property="hasFather">
<ruleml:var>x</ruleml:var>
<ruleml:var>y</ruleml:var>
</swrlx:individualPropertyAtom>
<swrlx:individualPropertyAtom swrlx:property="hasBrother">
<ruleml:var>y</ruleml:var>
<ruleml:var>z</ruleml:var>
</swrlx:individualPropertyAtom>
</ruleml:_body>
<ruleml:_head>
<swrlx:individualPropertyAtom swrlx:property="hasUncle">
<ruleml:var>x</ruleml:var>
<ruleml:var>z</ruleml:var>
</swrlx:individualPropertyAtom>
</ruleml:_head>
</ruleml:imp>
```

The example provides a scenario of relationships. If we translate it into plain English it will become as follow;

x has father y

y has brother z

x has uncle z

So far, the notion of context has been discussed with necessary details. In order to use the context, in a way to help the system carry out reasoning, a reasoning mechanism is required. In the next section, some of the reasoning approaches are discussed.

The next section provides an insight view of the rule-based system. It defines how a rule-based system works and what are the different components of a rule-based system. The proceeding section details the working of a knowledge base, which serves as the rules repository. Further, the working mechanism of rule execution cycle and match-resolve-act is discussed.

2.9 Rule-Based System

There are quite a few methods to design a system as an expert system which have reasoning capacity [70], including rule-based approach. Rule-based system (RBS) is a computational method which is widely used in various applications of artificial intelligence. Mainly it is used for problems having similarities with human reasoning or expert systems.

Rule-based reasoning and traditional rule engines have found critical applications in practice, though mostly for desktop environment where the resources, e.g., computation or memory, are abundantly available compared to

smartphone devices. Rules can be traced back to early production systems as a well known and popular way for encoding expert knowledge into rules. As mentioned before, the RBS has a significant role in the field of artificial intelligence for modelling human reasoning and problem-solving processes in a specific domain. Human reasoning can be closely defined in terms of the if-else statement; therefore, RBS becomes a natural selection when it comes to encoding a human expert knowledge [71]. Each rule can carry a minute amount of knowledge and backed with the facts from the environment; it acts closely as the human brain. Different studies have backed the RBS as a close solution to human reasoning and problem-solving. The rules act as long term memory while the facts are considered to work as short term memory [72, 73]. RBS is used widely in various software of different domains [74]. Within these domains, an RBS can deliver as a consultant, problem solver, an expert or a decision maker [73]. The best usage of RBS is applied to the systems where the solution of a particular problem cannot be achieved using a conventional programming, or an algorithmic approach may not provide a better solution. Every day we encounter rules in our daily lives. While driving a car, we stop at a red light, and when the light is green, we move along. Similarly, a professor takes a class at a given time as directed in the schedule. The students also attend the class at the same time. So basically these are rules that we are following, and rules are what governs our daily routine [75]. For every domain, we acknowledge rules according to the context of the domain. For example, just like ontologies example for Jaguar

discussed earlier. Rules defined for car sales may refer to Jaguar as a car while the rules defined for a zoo or wild animals may refer to Jaguar as an animal. Therefore there are a set of rules for a specified domain so that the context in which the rules are received and executed are known. The rules in the RBS can be regarded as IF-THEN-ELSE statements, where the IF holds the condition for checking. The rules comprise of two main parts; one is the Left Hand Side(LHS) or condition while the other part is Right Hand Side(RHS) or consequent. If the rule conditions are satisfied by the facts, then a rule is fired, which can result in a new fact derived or a result being given out. This whole process is composed of different components that work together to make an RBS.

2.9.1 Components Of RBS

RBS components can be defined in line with definition by [76] as

- Rule-base contains a set of rules. Every rule has a conjunction of conditions called the left-hand side of a rule, and the right-hand side contains the action or set of actions. These rules carry an abundance of knowledge encoded in the form of rules from an expert or business rules.
- The working memory contains the facts, which are used by the rules. When the process starts, the initial facts are applied, which are vital for the system startup. Over time, more facts are added/deleted from

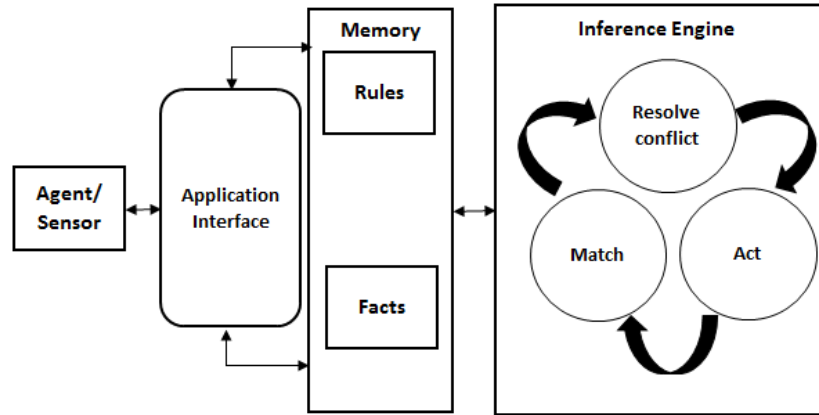


Figure 2.6: Rule Engine Components Logical Connection

the working memory causing certain rules to newly matching with the new facts and infer different results.

- The system execution is controlled by the inference engine or rule engine. The rule engine is mainly a procedure to match facts from the working memory to the rules in the rule-base. The efficiency of the whole system depends mostly on the design of the inference engine.

The schematic view in Figure 2.6 gives a logical linkage between the components defined, which shows the main idea of the RBS working mechanism. The advantage of RBS is that the rules are stored separately from the code. The rule-base can be altered without the changes in the program code. The rules syntax can be specified differently based on the different RBS platform. Despite the difference in syntax of a rule, execution is always carried in the same way.

2.9.2 Rule Execution

The LHS of a rule is composed of one or more conditions. In order to make the rule eligible to be fired, a rule is matched with the available facts. All the conditions of a rule must be satisfied. If more rules are eligible to be fired, then it is put into a conflict set. In the conflict resolution, a rule is chosen based on the criteria provided to fire a single rule from a conflict set. Conflict resolution can be determined by the rule priority, time stamp, latest rule to execute or combination of these or any defined criteria. This cycle is repeated unless all the rules are exhausted or a goal is reached. This cycle can be divided into three main steps, which are:

- **Match** The condition of a rule is matched with the facts in the working memory. Once a rule is satisfied, it is considered as activation of rule and provide an instance of the rule. A rule may have multiple instances. All rules which are activated are put forward for conflict resolution.
- **Select or Conflict Resolution** The conflict resolution strategy job is to select a single rule instance from a bunch of rules instances to be fired.
- **Act** Act takes consideration of the right-hand side of the rule. It can be a trigger to add/delete contents from working memory or to launch a subroutine. Once the rule is fired, the process is started again from the match phase.

The rules for a system has to follow a syntax; there are no specific guidelines for this and depends on the application being developed. For example, in some frameworks, a rule can have multiple RHS or actions, while some support only one action on RHS. The condition of the rule carries the knowledge part. The combination of rules into a rule-base makes a knowledge base. The knowledge base is iterated for pattern matching with facts, and that is the most expensive part of execution in terms of computation and time [74].

2.9.3 Conflict Resolution

The conflict resolution part comes after the rules are matched with the working memory. Initial facts decide which rules to be fired, but when more than one rule is fireable, then the inference engine has to decide which rule to be fired. There are quite a few strategies to handle the conflict, which depend on different scenarios and a different design on the conflict resolution mechanism. It can be merely a first rule to fire, random select, last rule to fire or may be based on some complicated method. Some RBS deploy different mechanism and let the end user select whichever method is required. Mostly, the recency is given priority where most recent facts are considered more, and rule matching the recent facts has to be fired [73]. Some other strategies include refraction and specificity. The refraction forbade any rule to fire again once it is already fired so that the other rules are given a chance to fire for the same set of facts. Specificity gives priority to rules which relatively have more conditions in a rule than others one.

2.10 Discussion

In this section, we have mainly focused on the basic terminology of the context and its related terms. The context being a building block of the whole framework is defined thoroughly and from a different point of views. Understanding of the context, context-aware and the systems based on these terms are vital for further understanding of this thesis. Therefore, different context based categories are discussed to broaden the idea and to understand how wide the concept of context-aware computing can reach. We also provided a flow of context from a sensor towards the application module and how it has to pass through a certain level in order to make context a meaningful piece of information. It is essential to understand the architectural style in which different context-aware systems are designed, as it gives the reader an understanding on why a system is designed in a certain way, where requirement plays an essential role in system design. Since context itself can be modelled in various methods, we defined the context modelling from the ontological point of view. Since the latter part of the thesis uses ontologies; therefore, it was deemed necessary to make the reader aware of the basic terms. Further, the context has to be used in a way so that the system can process it and take a decision on behalf of the user. Therefore the following sections of this chapter detail the rule-based systems. A rule-based system is not a new concept. It has provided a vast research area for scientist from decades. The ongoing research in AI and the expert systems further strengthen this

research area. Rule-base used on one system can be transferred into other rules format. Further, it is also a well-known strategy to use ontologies to obtain rules for some domain. Rules are easy to understand and write. It is quite easy to encode expert knowledge into rules. However, this calls for a thorough examination of a domain in which the RBS has to work. All the information has to be encoded in order to get an accurate output. Each rule contains a tiny amount of knowledge. Which makes it easy to add or remove some knowledge and also makes the rules independent from each other. While it also makes it hard to maintain the rule-base, especially when the size of rule-base is large. An RBS has a significant advantage of separate knowledge from the programming code. The knowledge as a whole can be reused, removed or replaced without the need to change the code. This yields to access an external part of the software, which can cause a delay. However, this delay can be reduced or eliminated with smart strategies so that the external part may be brought into a temporary memory, which is a part of the inference engine. Knowing the terms and terminologies explained in this chapter, it will be quite easy for a reader to understand the frameworks and applications which are based on current chapter reading. The next chapter details the available frameworks, rule-based engines and their combinations in frameworks and implemented applications. A well known RETE algorithm is also discussed in the next chapter which gives the user an understanding of pattern matching and the problem that can arise during the pattern matching phase.

Chapter 3

Related Work

3.1 Introduction

In the previous chapters, the notion of context-aware systems and a general structure of rule-based systems have been introduced. These two systems are two different areas in computer science with their independent research. In some of the research works, they have been combined to give a system intelligence by using the RBS and also make it context-aware, and we have also opted the same path. However, our focus remains on small devices instead of using servers, clouds computing or high-end systems. As discussed in Chapter 1, the authors in [7, 8, 9] argue that there is a lack of research for rule-based engine on mobile based devices. Our aim is not limited to mobile devices rather devices which has minimal resources. Such devices can be smart-phone or any such resource-bounded chip with support for

Android OS. We aim to provide a formal resource-bounded context-aware system development framework using rule-based reasoning techniques. The related work discussed here focusing on context-aware systems, RBS, or a combination of both. Different examples, platform, framework are discussed here for an elaborate study.

3.2 Context-Aware Development Platform

The frameworks discussed in this section describe the different approaches which use the different context modelling techniques along with the logic and rule-based context modelling techniques. These frameworks provide a general idea of their working mechanism, while in the later sections frameworks for resource-bounded devices focusing on the Android platform have been discussed.

3.2.1 Origin Model

The origin based framework based on the work of [77] is used for the development of context-aware systems in large scale pervasive systems. Examples include health care systems, management of the road traffic throughout a city, environmental monitoring and smart grids. Origins provide sufficient information to represent any context data, for example, sensors, web services databases, files or the combination of other origins. Origins provide the basic and backbone in the development of context-aware systems. The method to

retrieve the context from the origins uses the select and retrieve methods. It is already understood that it may not work as expected if the criteria for selecting a context is very strict. Generally, the four processing operations required by an origin are:

- Filtering
- Inference
- Aggregation
- Composition

Based on the origin model, the authors provide an origin toolkit with a proof of concept implementation which has been developed using SCALA programming language and the AKKA toolkit. The origin model defines origin as it can be any context source, and defines elementary parts in context-aware applications development. Origins type defines the nature of data an origin is associated with, for example, one origin may contain temperature, another origin may contain a city name, the composition of both provide relevant information by using the meta information of the origin. According to the authors, the origin should possess the following properties to qualify as an origin.

- Universal:** An origin should provide a universal interface to the context-aware application to access the information contained in its context source.

- Discoverable:** Origins have to be discoverable which can be based on the metadata and the information their respective context sources carry.
- Composable:** Different origins can be composed with each other to provide new context information.
- Migratable:** Origins can move from one place to another or from one machine to another in order to improve scalability and flexibility.
- Replicable:** More than one origin can carry information from a single source of context information, which can ultimately increase the reliability of the information received.

Summary of Origin Model

- Intended for the large, pervasive system.
- Origins are considered as context source, and are associated with the data source.
- Origins data can be combined.
- Implemented in SCALA¹ and AKKA² toolkit.

¹The Scala programming language

<http://www.scala-lang.org/>

²akka toolkit

<http://akka.io/>

3.2.2 ContextJ

In [78], the authors have provided a new approach to Context-Oriented Programming paradigm features by supporting an explicit representation of dependent context functionality that can be dynamically turned on or off. According to the authors, the context can be a variable, control flow property or an external event. ContextJ implements the layer in class style while classes contain their context-specific variation [79]. Previously the concept was provided by using LISP to improve the accessibility using ContextJ language. Context-oriented programming can be implemented using JAVA without any extensions to syntax/semantics. In ContextJ, a layer contains a layer identifier to refer a layer. Layers can extend classes except the final classes. Layer definitions contain partial method definitions. Method definitions can be distinguished as plain methods whose execution is not affected by layers. Method definitions consist of base method definitions which are executed when no active layer provides a corresponding partial method. ContextJ also provides a dynamic layer composition for a scoped layers activation. Scoped layers are controlled by a *with* block statement, which provides an argument list composed of layer identifiers that have to be activated. The *with* statements can be nested like any Java block statements. Opposed to the *with* block statement there is also a *without* statement that deactivates a scoped layer. It is mainly intended to stop the interference of layers that may be initiated by partial definitions of various methods.

Summary of ContextJ

- Explicitly represent dependent context functionality.
- Context can be a variable, property or any external event.
- Implement layers in class, layer identifier identify its layer.
- Use Java programming libraries without any extensions.
- Introduction of *with* and *without* properties to activate and deactivate certain layers.

3.2.3 JCAF

JCAF [49] is a framework and programming architecture for context-aware applications. JCAF is service oriented, distributed, and event-based secure infrastructure suitable for the development and deployment of a wide range of context-aware applications. It also suggests a compact Java API, which can be implemented and extended in special purpose context-aware systems. The main goal of JCAF is to devise a lightweight framework based on JAVA with a small set of interfaces. It is intended for researchers and innovative development. Although distributed but JCAF can be used in centralized systems. JCAF is composed of a programming framework via API and runtime infrastructure; the core design principles of runtime infrastructure are to provide a distributed and cooperative service system. In the JCAF the main modelling concept is the entity, an entity has a context, and the context further carries the context items and these all are available in its respective Java interfaces which a programmer has to implement in order to use them.

In JCAF, a context-aware system should be event-based infrastructure with the ability to react to changes in its environment. Context-aware systems developed in JCAF's security and privacy should be protected, and it should have access control on different levels. The key to the context information in JCAF is its origin credibility. It is recommended in JCAF that extensibility may be provided without restarting the services and any deployable module may be added without interrupting the context services. It should support the evolution of supported types of context by dynamically loading context definitions. The main design principles of its API are semantic free modelling, context quality and support for activities.

Summary of JCAF:

- Developed in JAVA.
- It is service oriented, distributed and event-based infrastructure.
- Can use compact Java API for special purpose systems.
- Can also be used in a centralized system.
- Main design objective are semantic free modelling, context quality and support for activities.

3.2.4 OPEN

The authors of [80] present an ontology-based programming framework for rapid prototyping of context-aware application development. The design goal is to support a wide user's category, cooperation and collaboration in the

application development process. The framework further emphasizes that being a collaborative environment, users have to agree on a shared conceptualization of the domain. The authors also targeted three categories of users based on their technical abilities into High-level, Middle-level and Low-level users who can use the framework in a different environment. The framework, while supporting collaboration and sharing of context, also focuses on the cooperation between users. This cooperation can be synchronous, asynchronous, individual and group based. The cooperation pattern based on the technical abilities can be between developers, developers and end users, and between end users. The main components of the OPEN framework are context providers, the context manager, programming toolkits and the resource sharing server. The framework although has various options to cater users from diverse technical skills, however, the use of resource sharing server suggests a limitation on distributed approach, and also the Android limitations demands a more compact and Android compatible framework.

3.3 Rule-Based Systems And Existing Rule Engines

3.3.1 Prolog

Prolog³ stands for programming logic, developed as horn-clause based declarative programming language. Rules and logical structures are used for programming in Prolog to provide the relation between objects, represented as facts and rules. A query then initiates a computation on these rules to get the results. In Prolog, a programmer has to specify a goal to be achieved, and the Prolog system works it out on how to achieve it. Prolog can be used in distributed environments. The authors of [81] have proposed an idea of extending Prolog by using DAHL extension. The approach uses Prolog DAHL extension and a C written networking backend. The message passing is carried out by the nodes sending messages to connected nodes or directly to the root. The messages received are used as a local query and can be processed locally.

Prolog is a widely used tool for rule-based system development, and some areas where Prolog is used are for expert systems, machine learning and intelligent computing.

Summary of Prolog.

-Based on horn clause rules.

³SWI Prolog
www.swi-prolog.org

- Widely used and a mature declarative language.
- Supports backward chaining.
- Programmer specify the goals.
- Can use DAHL extension for networking in special requirement.

3.3.2 Jess

Jess⁴, is a Java-based declarative programming tool. It follows horn-clause based declarative language, and popularly known as the rule engine for the Java platform. It takes the rules as an XML or the Jess rule language. Its native language of Java enhances the power of Jess. Java provides a wide collection of libraries that can be used in Jess. Jess programming can be done in Eclipse⁵ or any similar Java Integrated Development Environment(IDE). Using the Jess engine, a developer can develop applications that can reason using the knowledge supplied in the form of declarative rules. As the language uses the Java Virtual Machine, it can be used on multiple platforms. Jess can also be used in a distributed environment. In [82], one of the distributed approaches to use Jess has been discussed; however, the internal mechanism used in the development is not very clear. Also, it does not elaborate on the connection strategies. The mechanism used for invoking transmission is also not defined.

Summary of Jess:

⁴The Rule Engine for the Java™ Platform <http://herzberg.ca.sandia.gov/>

⁵Eclipse <https://eclipse.org/>

- Based on Horn Clause rules.
- Java-based declarative programming tool.
- Use XML based rule as input or in JESS format.
- Used Java Virtual Machine which provides portability on desktop platforms.
- Can be used in a distributed environment.

3.3.3 Clips

CLIPS⁶ is an expert system development tool, and one of the most widely used tools for the said purpose. It was developed at NASA as its expert system replacement. CLIPS is written in C language, and CLIPS stands for C Language Integrated Production System. CLIPS incorporates the various programming strategies such as OOP, procedural, rule-based and logical. The CLIPS interpreter for rule-based languages has four steps. In the first step, it matches rules with the antecedents. When the combination of facts satisfies the rule, it is known as instantiation, and each matching rule is added to the agenda. The next step resolves the conflict. It selects a rule for execution from the agenda; if there are no rules to execute it, then goes to the halt state. In the third step, the rule is executed, which has a specified action to perform. The fourth step does not do anything; it simply repeats the process and goes back to step one. CLIPS consists of a database which is a list of facts. Each fact contains one or more than one fields enclosed in

⁶A tool for building expert systems <http://clipsrules.sourceforge.net/>

brackets. They represent information and are used to define the problem's current state. This database is also known as declarative knowledge. For example the rule ($< XY$) shows that the values of X is less than Y . The first field represents the relation between the two remaining fields. Some examples of facts are

(relation patient "patient of Dr John")

represent the relation of a patient with the Doctor named John

(Patient Diabetes)

represents that Patient has Diabetes

(Patient Diabetes Hypertension)

represents that Patient have Diabetes and Hypertension

("Patient has Diabetes and Hypertension and needs treatment")

represents a long string usage as a single fact.

The working mechanism of CLIPS is simple, and it loads information from a file into memory storage where memory is composed of storage and rules.

Summary of Clips.

- CLIPS is an expert system development tool.
- Developed in C language.
- Used by NASA for their expert system.
- Process rules by matching, conflict resolution, execution and restart.
- Easy to understand rule format.

The frameworks discussed in this section are purely context-aware or rule-based systems that are designed mostly for high-end systems. Next, we

discuss some frameworks and applications that are designed for the mobile platform.

3.4 Rule-Based Systems For Mobile Devices

In our study, it was discovered that a considerable amount of work is based on social networks. In a social network, a user puts his/her lot of details, preferences, likes and dislikes related to him/her. These give a considerable amount of context related to a user, as discussed in a different research project such as SociaCircuit platform[83] which monitor different social factors between the users. Based on these factors, it measures the shift in user preferences, e.g. habits and behaviours. The work discussed in [84] focused on finding social relationships between the users; this provides results based on some data mining tools. Sociometric badge [85] monitor employees different activity patterns in the office. It records different data related to the user and based on that data, within the organization, user's job satisfaction and interactions quality can be predicted. Similarly, [86] also monitors a user activity based on his/her different mobile sensors, his/her locations visited and call logs as an example. This monitoring then further try to infer the important location based on his/her social activities, different relationships and related information. Recent work based on inferring results or mobile based expert systems still lacks different aspects. For example, in [87], a small expert system is developed, which acts as an academic advisor. It has

some set of rules which takes six inputs from the user and based on this input advise accordingly. The system is monotonic, as it will give the same answer for the same inputs every time, there is no any capacity to run a different set of rules as the interface is also linked with its own current set of rules. In some of the research work, a workaround is done by making a system in a client-server architecture such as [88] where a server is working as a knowledge base, and Android phone is working as an agent with an application installed to connect to the server and send some context, e.g., location. Similarly, another research work [89] although not for Android instead developed for iPhone platform uses the same client-server architecture combined with the rule-based system on the server to provide a safe evacuation in case of emergency cases at a university (case scenario). However, the set of rules used as expert knowledge is not defined in their work. Some attempts have been made in order to bring the already mature expert systems into the Android platform if not all, e.g., iOS, Microsoft mobile, which are well known mobile platforms. Android uses Java, but it lacks some classes that can only be used in the desktop environment and not on Android. Some examples of portability are defined along with the issues faced while in the porting stages. In a recently published book for smart applications [90], detailed insight is given to rule engines that can be used on Android. However, they are not resource friendly, or context-aware, nor they use preferences for dynamic adoption of context. The *Jruleengine* and *Zilonis* has issues with the use of some operators, e.g. OR operator. In the *DTRules*, the facts are provided

as an XML file, which makes it less suitable in systems where the facts are provided in runtime. In the *Termware*, the rules are written in the code, hence making it tedious to update the rules. *Roolie*, suffers from redundant files for rules, as each rule needs to be saved in different JAVA file. Aside from these, there are also technical issues involved [91] such as in *Drools*, it consumes much memory and eventually runs out of memory while converting to Dalvik format. *JLisa* throws a stack overflow error while running on Android. *Take* requires JAVA compiler at runtime, not Android compiler. *Jess* porting is not advisable as the developer license cost around USD 15000; also, it does not have any compatible version for Android. *OpenRules* uses some of the JAVA classes that are not available in the Android environment; besides these classes, it also consumes a lot of memory.

To the best of our knowledge and the study of the literature review, it is a known fact that majority of the rule engines run the RETE algorithm on their back-end as compared in Table 3.4, such as JESS. The Android itself, as stated earlier, is not limited to the smartphone only. Android can be installed on embedded chips and TV beside other small devices, which makes it a very suitable platform capable of running a variety of applications, and that makes it a suitable operating system for our research. To provide a context-aware rule engine on a resource-limited device, an alternative to the existing technology is required which must be lightweight, efficient and resource friendly. The system discussed in this Chapter and compared in Table 3.4 are lacking at least one of the following major issues such as

- Context re-usability
- Generic modelling
- Not For resource-bounded device
- Not native to platform
- Context-awareness
- Porting issues
- RETE based

Regarding the issue of re-usability of a context, some frameworks provide ontology-based approach such as in [92, 93]; however, they do not address the issue of context-aware mobile application development. Some recent work has effectively used the ontologies for modelling with better resource handling. They have modelled their system from the ontology, with the bound on resources such as memory and communication [67, 94]. The typical problems of RBS combined with context-aware systems, re-usability of contexts, low resource usage and others, as discussed in this section, has provided us with an opportunity to explore further the problem on small devices. Moreover, to devise an algorithm and context-awareness model that can perform in comparable computation and with better memory usage along with using the contexts that a smartphone can provide. In order to achieve this, first, the introduction of the current state of the art RETE algorithm is necessary and to analyse how it works.

Name	Context- Reusability	Generic	For Re- source Bounded Devices	Native	Context- Aware	Porting Is- sues	RETE based
socialCircuit	Y	N	N	N	NA	NA	NA
FMES [87]	N	N	N	Y	N	NA	N
KBAM [88]	N	N	N	Y	Y	NA	N
ASE [89]	N	N	N	Y	N	NA	N
Clips	NA	Y	N	N	N	Y	Y
JRuleEngine	NA	Y	N	N	N	Y	Y
Zionis	NA	Y	N	N	N	Y	Y
JEOPS	NA	Y	N	N	N	Y	Y
JxBRE	NA	Y	N	N	N	Y	N
Drools	NA	Y	N	N	N	Y	Y
JLisa	NA	Y	N	N	N	Y	N
Take	NA	Y	N	N	N	Y	N
OpenRules	NA	Y	N	N	N	Y	N
Proposed	Y	Y	Y	Y	Y	NA	N

Table 3.1: Comparison Table For Rule Engines. (Y: Yes, N: NO, NA: Not Applicable)

3.5 RETE Algorithm

RETE [95] algorithm is widely used in systems, where pattern matching is required, such as a rule-based system. Charles Forgy introduced RETE as part of his doctorate studies. Based on the RETE algorithm, several other algorithms were developed for high-end computers. It may be noted that the RETE algorithm is used widely on centralised systems. Although, computationally improved but still consume a huge amount of memory [72] especially when it comes to the execution on single device or small devices [71]. There have been some attempts to port the current RBS systems [96, 88] into Android platform with little to no success, which has been discussed in survey work [97], including JESS which is based on RETE algorithm ⁷. RETE is no doubt one of the algorithms which are commercially used in large corporates and where the business rules are in large numbers. Before going into details, we introduce the basic terminologies of the RETE algorithm. It considers the production memory (PM) and working memory (WM). PM contains different productions or rules. Each rule is represented as a set of conditions on the LHS and its respective actions on the RHS. WM contains items which represent facts. The structure of a particular rule is provided below

```
(  
  name of the rule
```

⁷<https://stackoverflow.com/questions/44924473/how-to-use-jess-in-android>

Left-hand-side , i.e. with one or more condition

—>

Right-hand-side , i.e. with one or more actions

)

Generally, the matching algorithm ignores the action part. Rather it only handles the conditions. The rule conditions may contain constants and variables. So the matching part's job is to match the LHS with the facts and if there is a variable, then bind it with its corresponding value from the facts. If all the conditions are satisfied, then add it to the conflict set. The Actions are taken care of by another part of the system once a conflict set is created. RETE algorithm makes use of dataflow algorithm for the rule conditions presentation. The network can be further broken down into two main parts, namely the Alpha part and the Beta part. The Alpha part carries out the constant tests on the working memory elements and stores the results in the Alpha memory/memories. This Alpha memory contains the elements of the working memory, which successfully passes the constant test for a given condition of a rule. The Beta part handles the joins and beta memory/memories. It does the necessary variable binding between conditions, and store the results in the join node. Beta memories then store along with the semi-matched production rules, as more and more steps it takes. The process is repeated for the rest of the conditions, and finally, a fully matched production is acquired. Changes in the working memory are conveyed to the

Alpha network, and related Alpha nodes adopt the changes. Ultimately these changes are passed to the Beta network nodes and joins. Any new matches if found in the Beta network is updated accordingly till it reaches the end. At the end of the network, we have the production node, when a production node is produced, it indicates that a newly matched rule is found. In the mid of the process, we have activations of two types left activation and right activation. Left activation corresponds to the activation of any node by any other nodes in the Beta network. While the Right activation refers to the activation of any node by the Alpha memory. The joins in the Beta network can have these two types of activations. Both activations are handled by different procedures and are discussed in the analysis of the algorithm section. One important feature of the RETE algorithm is its state saving. It saves the states of the matching process in the Alpha, Beta memories. With a change in the WM, many nodes are not affected. However, the RETE algorithm is not recommended for systems where major changes occur in the working memory [98]. Another feature of the RETE algorithm is its node sharing with productions with similar conditions. Single Alpha memory is used for a rule which has the same common conditions. The Figure 3.1 gives a logical network illustration and the left and right activations.

3.5.1 Analysis Of RETE Algorithm

In [98], an algorithm, based on RETE, was published and points out that RETE slows down with an increased number of production rules. Further-

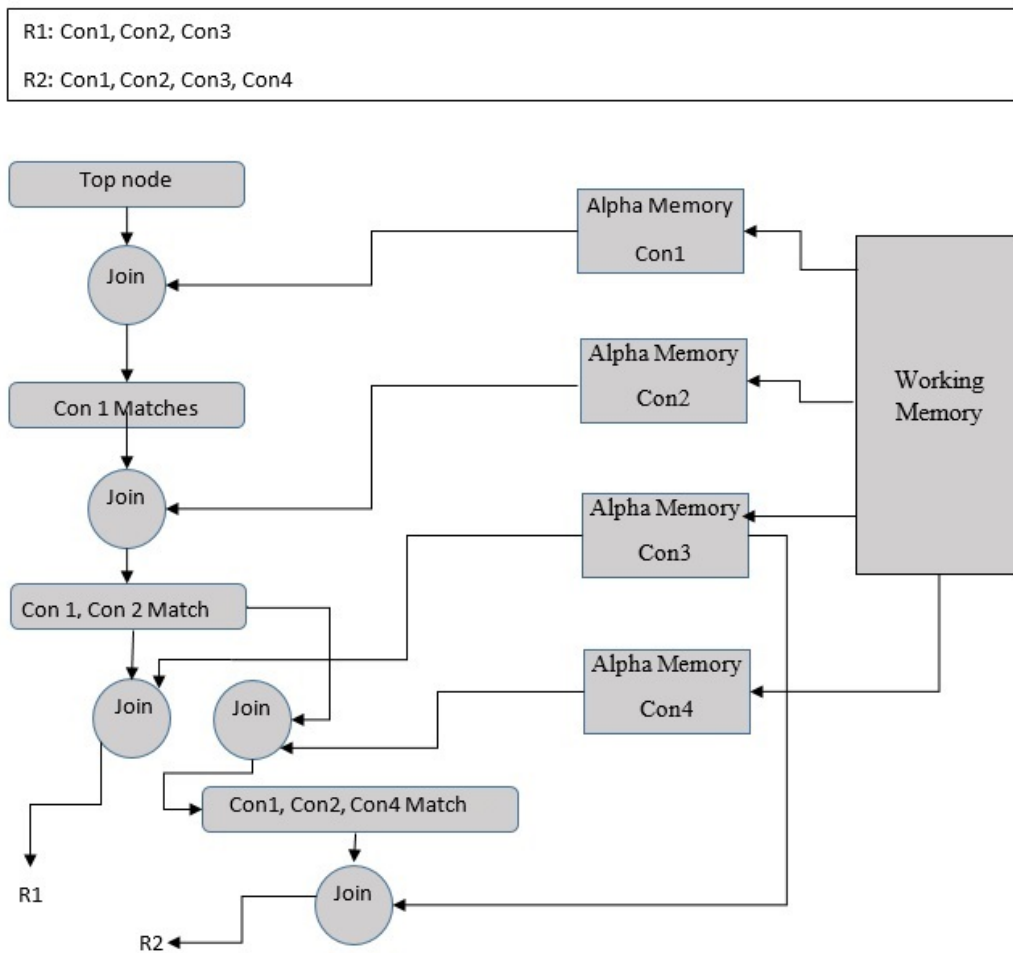


Figure 3.1: RETE Network Illustration

more, the RETE algorithm is not designed for a system where the working memory (WM) is frequently updated. Moreover, The RETE algorithm worst case can reach to $O(WM_e^{RC})$ where WM_e refers to the working memory elements while the RC means the number of conditions in rule [71, 99]. In the context of multi-agent systems, the memory consumption of the RETE algorithm is problematic [71]. Since the working memory is also not fixed, and RETE algorithm is well known for its large use of memory it is not a suitable candidate for small devices especially with a tiny memory. It uses a lot of memory when creating a Beta network and Alpha network and the space complexity is exponential [100] for both RETE and its similar algorithm TREAT. Other issue is when there are many WM elements and a complex rule with varying conditions. This can lead to the cross product problem and potentially can take the system into worst case. Further, an issue with the RETE algorithm as pointed in [101] is the creation of lot of child nodes when we have attribute with multiple values e.g., colour and its values . In that case the attribute colour node will spread into the number of values (blue,green,black and so on) available. Beside these problems pointed out by different research, if we consider the pseudo code of the RETE algorithm and analyse the complexity we found that the Left and Right node activation (Table 3.2 and 3.3) has an asymptotic complexity of $O(n^2)$. Considering the (Join Node Left activation and Join Node Right Activation) Pseudo code from [98] for RETE algorithm.

Algorithm	<i>cost</i>	<i>frequency</i>
START		
IF node.parent just became non empty then	c_1	n
relink.to.alpha.memory(node)	c_2	n
If node.amem.items =nil then	c_3	n
remove node from the list node.parent.children	c_4	n
For each item in node.amem.items do	c_5	n
If perform-join-tests(node.tests,t,item.wme) then	c_6	n
For each child in node.children	c_7	n^2
do left-activation(child,t, item.wme)	c_8	n^2
END		

Table 3.2: Algorithm Left Activation Running Cost

Algorithm	<i>cost</i>	<i>frequency</i>
START		
IF node.parent just became non empty then	c_1	n
relink.to.beta.memory(node)	c_2	n
If node.parent.items =nil then	c_3	n
remove node from the list node.amem.successors	c_4	n
For each t in node.parent.items do	c_5	n
If perform-join-tests(node.tests,t,w) then	c_6	n
For each child in node.children	c_7	n^2
do left-activation(child,t, w)	c_8	n^2
END		

Table 3.3: Algorithm Right Activation Running Cost

These two sections particularly have the complexity of $O(n^2)$. While the rest of the pseudo code, itself is out of the scope of this thesis to be discussed here.

3.5.2 The Match Problem

According to Charles Forgy, the matching phase can take up to the ninety per cent of the whole execution time [102]. The matching phase is repeated numerous times and repeated when new working memory elements are added or removed. It certainly has a significant impact on the overall execution time. The matching time is effected with the number of conditions in the rule and the number of working elements. As in each rule, we have to match the condition with the facts in order to satisfy any condition, so the time for execution gets longer when there are many conditions in a rule. Other factors that affect it further can be attributed to the number of variables on the LHS of the rule. If there are many variables, and some variables are repeated in other rules. It should be binding to the same fact every time. Semi matching rules also create a problem as they are not added to the conflict set. However, they are tested for qualifying the facts. Rules that are never fired are also checked for eligibility. Long rules with many conditions also create problems and are called the long chain effect. These are some of the frequently occurring problems. There are instead precautions than solutions to avoid the matching problem. The precautions may include saving the state of the rule conditions, keeping track of facts in view of the rule by

keeping track of the rules that are most probably affected with the change in WM and sharing conditions of rules with similar rules. However, these precautions/solutions have their drawbacks. As we already mentioned that most of the expert system research is mainly on the high-end computer with a lot of resources, and the solutions to match problem take advantage of using the abundant resources. As the state saving, condition saving and similar strategies take a lot of memory. While our concern is to avoid such issues and deliver comparable or better results on small devices which naturally have small memory size. Different methods can improve the match, the rule being the main components can drastically improve the overall performance. Simple ordering conditions in a rule has an effect. If a rule has ten conditions and the first nine conditions match while the last one does not match, then the rule is not eligible for firing, and this check wastes the resources for matching the nine conditions. Instead, if the tenth condition is at first place, it will save much computation resources. Researchers have identified several rule conditions for improvement and can be found in [72, 103], which include strategies such as sharing condition, rule ordering, facts ordering besides other strategies. These can be carried out while in the design phase of the rule-base.

These issues accumulate to demand a concrete solution. Our proposed framework, discussed in the next chapter, covers all such issues and make it possible to run such an expert system which is native to the mobile platform and provides satisfying results.

3.6 Discussion

In this chapter, we have sequentially defined technologies based on the understandings from the previous chapter. We independently and generally discussed different models from context-aware systems, rule-based systems and then their usage on a mobile platform. The reason for such a discussion is to understand the applications of such concepts. Generally, context-aware systems can be with or without a rule-based system and vice versa. However, the recent advancements in mobile technologies have made it quite favourable to employ the sensors on the mobile device to perceive its environment, while the rule-based technology uses the same sensor information and convert it into processable context. The reason to explain them separately is to give a reader a clear understanding of such frameworks/technologies. The merger of these technologies can provide a framework with two main properties. First, to perceive the environments using the context-aware terminology and secondly, rule-based systems come handy to process these context and reason about them and give some output. As these technologies work mainly on high-end computing devices, especially the reasoning part. There is a gap for development of such systems on low-end devices such as mobile phones with certain limitation on resources as discussed. Therefore, our main target for development is Android-based devices, in the Section 3.4 we further discussed the applied technologies on mobile devices and briefed along with issues that have to be avoided in order to make a resource-friendly framework. The prob-

lem is not using the currently available frameworks is the portability issues that arise when these frameworks are ported, as discussed in Section 3.4. In order to avoid such issues, it is necessary to make the framework from scratch and as a native application framework to ensure the compatibility with the proposed platform. As the Android itself is evolving very fast, it is easy to update the native code then the ported code, as the Android SDK is getting mature, it is deprecating many classes, methods to be replaced with newer ones. Which can be problematic for the ported code. Another main problem associated with implementing, especially the rule execution part on mobile devices is the pattern matching algorithm, i.e. RETE, which is known as a resource-greedy algorithm. While the resources are limited in our case, we detailed the RETE algorithm in Section 3.5, defined its working mechanism, complexity and some problems associated with the RETE algorithm. The reason to discuss these all technologies separately gives a reader better understanding of different parts of the framework which we have proposed in the next chapter, where all these parts are integrated, and our custom-built algorithm is used with better memory and space complexity.

Chapter 4

Proposed Platform, Context Acquisition Scheme And Algorithm

4.1 Introduction

In this Chapter, the proposed platform is described based on which the whole framework works. The context acquisition model is also defined, followed by the core components of the system, along with the proposed algorithm.

4.2 Proposed Platform

For the test cases, we chose the Android platform. The selection was based on various facts. Android is the largest mobile operating system with huge market dominance. The languages used for the context-aware systems development, to name some, are JADE¹, JARE², JESS[82] and many more [104] use JAVA for their frameworks. JAVA being platform independent can be easily ported to any platform, which has the supported Java Virtual Machine (JVM). The only downside of using JAVA is that it has some compatibility issues with mobile devices platforms such as Android. The main language for Android development is Google's JAVA using the official Android Software Development Kit (SDK) [105]. There are differences between Oracle JAVA programming language which is used for desktop systems and Google's JAVA programming for the Android systems. As for the JAVA, the syntax of the language remains the same. The basic difference lies between their low-level machine code generations. The desktop systems use JVM to translate JAVA code into machine-readable code or bytecode while in Android system the application is executed using the Dalvik Virtual Machine [106]. Dalvik VM is a compact VM and is used in resource-bounded devices. The JAVA language which can be used for Android does not support all the classes of JAVA. The reason behind it is that it has been optimized for the use with the resource-

¹Java Agent Development Framework <http://Jade.tilab.com>

²JARE- Java Automated Reasoning Engine <http://faculty.cs.tamu.edu/ioerger/Jare/Jare-old-page.html>

bounded devices. It is a good practice to develop the whole framework from scratch using Google Android SDK, so that no compatibility issues arise.

There are also other platforms exist, including iOS by Apple Inc and Microsoft Windows Mobile, but in order to reach the maximum devices, it is clear from an online article published in StatCounter.com³ that the Android has the maximum user base, compared to iOS and others. However, it does not limit the research objective to Android only, and there will be a tendency to make a context-aware framework that can run on all platforms seamlessly in the future.

4.3 Context Acquisition Model

Context categorization schemes allow the context to be understood in its operational or conceptual limitations and factors. The conceptual categorization defines the conceptual relationships between contexts. It provides a wide guide for the identification of the related context, while the operational categorization scheme deals with the problems and issues related to context acquisition techniques, which may include information about context source, frequency, and methods for data acquisition. The proposed solution is operational as the context will be acquired from the physical sensors in static form, and according to the context, data will be processed further. Table 4.3

³<http://gs.statcounter.com/os-market-share/mobile/worldwide>

summarizes the different aspects, followed by explained commentary on each terminology.

4.3.1 Context Acquisition

The context acquisition refers to the methods by which a context can be acquired from the source. The source can be agents or sensors. The two methods generally used are the *pull* and *push* methods. Pull tends to acquire the data from the software of the sensor. The software decides the acquisition of context and communication. It consumes bandwidth as the software has to send the request to the sensor to acquire the context. While in the push method, the sensors independently sense the data and send it. The communication decision is taken by the sensor itself, which can be pre-programmed. The push method is recommended for the proposed system, as the sensors continuously sense its environment and send any changes in the context.

4.3.2 Context Acquisition Frequency

The frequency of acquiring the context from the sensor can be *instant* or *interval* based. Instant sends the data whenever certain conditions are met, e.g. rise in temperature is sensed. While the interval sends the data after a determined amount of time is passed. Instant saves energy as it may not send data unless it needs to, while the interval has to send data regardless of

	Operational Conceptual	Push Pull	Instant Interval	Direct Middleware	Physical virtual logical	Key-value Markup Schema Graphical Object based Logic Based Ontology Based	Supervised Learning Unsupervised Rules Fuzzy-logic Ont-based
Context Catego- rization Scheme	Operational						
Context Acquisition		Push					
Frequency			Instant				
Sensor Ac- cess				Both			
Sensor Type					Physical		
Context modelling and repre- sentation techniques						Markupscheme, Ontology based and Logic based	
Context reasoning							Rules

Table 4.1: Summary of proposed system

any change in context. The proposed system will use the instant approach, but the intervals can also be set in order to consume less energy.

4.3.3 Context Acquisition Methods From Source

Context acquisition methods can be of various types; it can be direct, through middleware or a context server. As the system has to be distributed, and the servers are not used, the context server cannot be used. The remaining two methods can be used in a distributed approach. As discussed in context acquisition that push is the recommended context acquisition technique, the direct sensor access will provide excellent support for the push method. The middleware can also be used where the direct sensor approach is not possible, e.g. acquiring context from another agent. As our proposed system has to acquire context from the sensor itself, which can be directly accessed from the sensor, while it can also get the context from other agents which are software-based, so both approaches have to be addressed.

4.3.4 Context Reasoning

Context reasoning is the main part of the system, which will infer the context and provide output. There are a variety of reasoning techniques, which vary from each other on certain factors. Context reasoning may be under supervised learning, unsupervised learning, rules, fuzzy logic, ontology-based and probabilistic approach as dominant approaches as discussed in [24]. Each

has been discussed with advantages and disadvantages in the same reference. The proposed system is based on the rule-based system.

4.4 Model Of Context-Aware Systems

We model context-aware systems as a multi-agent defeasible reasoning system [107]. In this model, the incomplete or inconsistent context information is dealt with the non-monotonic reasoning, while non-monotonic reasoning is when the conclusion of logic can be invalidated by adding more knowledge. The proposed model is designed for the resource-bounded environment. The resources mainly considered here are memory, time and computation.

More formally, according to [107] a defeasible theory \mathcal{D} is a triple $(\mathfrak{R}, \mathcal{F}, \succ)$ where \mathfrak{R} is a finite set of rules, \mathcal{F} is a finite set of facts, and \succ is a superiority relation on \mathfrak{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 $\sim P$. For example, given a literal l , the complement $\sim l$ of l is defined to be P if l is of the form $\sim P$, and $\sim P$ if l is of the form P . In the rules, we assume variables are preceded by a question mark and constants are plainly defined. In \mathcal{D} , there are different kinds of rules those are often represented using various arrows. However, the proposed framework mainly use two types which are discussed below.

Strict rules are of the form : $P_1, .P_2, \dots, P_n \rightarrow P$ where the conclusion P is

valid whenever its antecedents P_1, P_2, \dots, P_n are true. An example of a strict rule can be “A person who has a patient identification number is a patient”, which can be written as $\mathbf{r1}: Person(?p), PatientID(?pid), hasPatientID(?p, ?pid) \rightarrow Patient(?p)$.

Defeasible rules are of the form: $P_1, P_2, \dots, P_n \implies P$ and they can be defeated by contrary evidence. An example rule can be $\mathbf{r2}: Patient(?p), hasFever(?p, High) \implies hasSituation(?p, Emergency)$. This rule states that if the patient has a high fever then there are provable reasons to declare an emergency situation for patient, unless there is other evidence that provides reasons to believe the contrary. For example, a defeasible rule $\mathbf{r3}: Patient(?p), hasFever(?p, High), hasConsciousness(?p, Yes) \implies \sim hasSituation(?p, Emergency)$. We can observe that the defeasible rule $\mathbf{r3}$ is more specific (we assume that $\mathbf{r3}$ is superior to $\mathbf{r2}$ i.e., $\mathbf{r3} \succ \mathbf{r2}$) and it could override the rule $\mathbf{r2}$. That is a defeasible rule is used to represent tentative information that may be used if nothing could be placed against it.

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 $Tell$. $Ask(i, j, P)$ means i asks j whether the context P is the case and $Tell(i, j, P)$ means i tells j that context P ($i \neq j$). The positions in which the Ask and $Tell$ primitives may appear in a rule depends on which agent’s program the rule belongs to. Agent i may have an Ask or a $Tell$ with arguments (i, j, P) in the consequent of a rule; e.g., $P_1, P_2, \dots, P_n \rightarrow$

$Ask(i, j, P)$. Whereas agent j may have an *Ask* or a *Tell* with arguments (i, j, P) in the antecedent of the rule; e.g., $Tell(i, j, P) \rightarrow P$ is a well-formed rule (we call it trust rule) for agent j that causes it to believe i when i informs it that context P is the case. No other occurrences of *Ask* or *Tell* are allowed. When a rule has either an *Ask* or a *Tell* as its consequent; we call it a communication rule. All other rules are known as deduction rules. For the rule execution, there is a well-defined procedure for rule execution, which includes the rule priority, rule selection strategy, available actions and effects of the action. However, the rule must have a format for a given framework for seamless execution.

4.4.1 Rule Format

There is no standard format for writing rules. Which gives designer flexibility of usage. In our proposed model, we have embedded some extra information with the rules so that it can be efficiently used. A standard rule format for our framework is written as $m : P_1, P_2, \dots, P_n \rightarrow P_0 : F$. while m represent the priority of the rule followed by the LHS and RHS. The F is a flag which indicates the nature of the rule either communication, deduction or goal. Similarly, as we will further study preferences in the next Chapter, we also embed an extra field of CS . CS stands for context set, which works as an indicator for a particular rule. The indicator indicates if a rule belongs to any particular subgroup of rules which is used for the preferences purposes. A rule format of our framework with preferences incorporation looks like: m

: $P_1, P_2, \dots, P_n \rightarrow P_0 : F : CS$ where $n \geq 0$. The same rule format is used on different agents to solve problems in a distributed fashion.

4.4.2 Context-Aware Systems as Resource-Bounded Agents

The main role of multi-agent systems research is distributed problem-solving (DPS). Such an approach allow the agents to solve a particular problem collaboratively. According to Smith and Davis “*distributed problem solvers offer advantages of speed, reliability, extensibility, the ability to handle applications with a natural spatial distribution, and the ability to tolerate uncertain data and knowledge. Because such systems are highly modular, they also offer conceptual clarity and simplicity of design*” [108]. However, even in DPS approach, the resources consumed by an agent are the focus of interest. Our framework has a constraint on various resources because most of the time, context-aware systems are deployed on small devices, e.g., phone or remotely deployed sensors, with minimal resources available for their operations.

To clarify the DPS mechanism, let us present a basic example of a DPS using two agents. Agents reason using (Horn clause) rules and communicate via messages with each other. The knowledge bases and initial working memories of agent 1 and agent 2 are shown in Fig. 4.1. The goal is to derive context $C_5(a)$. Note that in the rule $R_{ik} m : body \rightarrow head$, R_{ik} represents k^{th} rule of agent i and the number m represents annotated priority of the rule.

Every transition corresponds to an execution step and transforms an agent

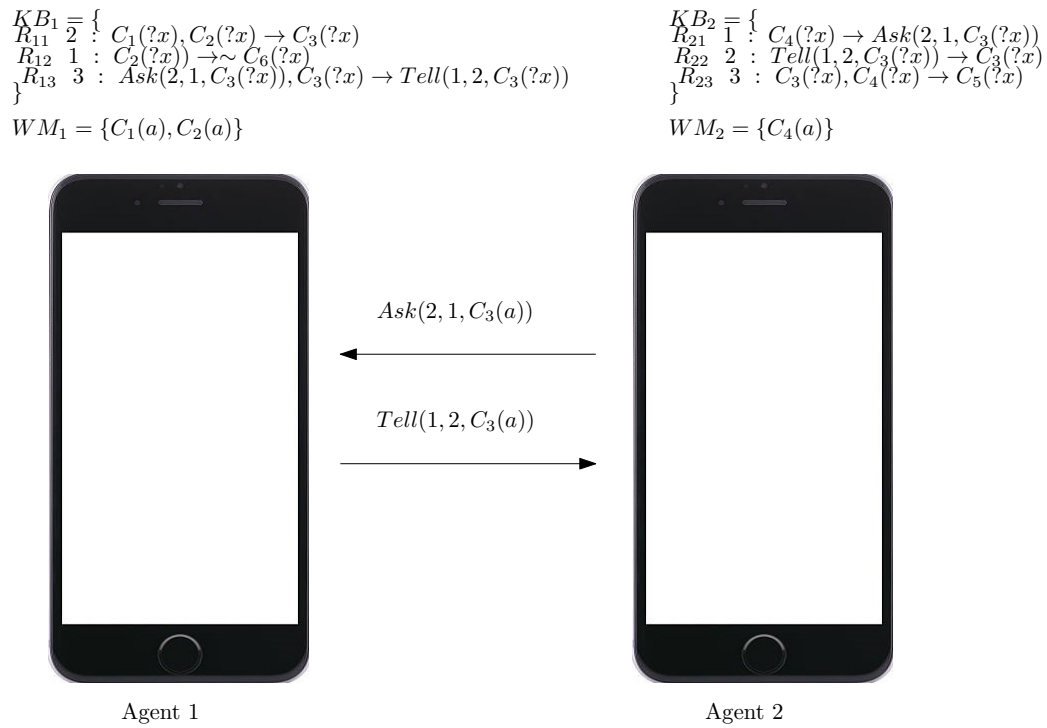


Figure 4.1: Distributed problem solving

from one state to another. States consist of the rules, facts, contexts. A *step* of the whole system is composed of the actions of each agent, in parallel. If any agent at any state solves a problem, it makes the system to reach its goal. An example run of the system is shown in Table 4.2. In the table, a newly inferred context at any given time is shown in blue text. For example, antecedents of rule R_{11} of agent 1 match the contents of the memory configuration and infers new context $C_3(a)$ at step 1. An overwritten context is shown in red text, and a context which is inferred in the current state and which will be overwritten soon is shown in the cyan text. As far as memory is concerned, static and dynamic memories are separated using | where the left side represents the static part, and the right side represents its dynamic part. The size of the dynamic part of agent 1 is 2 units, and that is of agent 2 is 1 unit. As per the framework mechanism, once rule conditions fully qualify for activation, only then it can be fired. There can be instances when multiple rule instances are eligible to be fired, and such case will be handled using the priority of the rule. It is evident that in Fig. 4.1 neither agent can derive (infer) $C_5(a)$ alone. We can observe in Table 4.2 that the resource requirements for the system to derive the goal context $C_5(a)$ are 2 messages that need to be exchanged by each agent and 6 time steps. We can also observe that, if we reduce the dynamic memory size for agent 1 (and for agent 2) by 1, then the system will not be able to achieve the desired goal. The example here is a very simple case; however, detailed execution along with preferences are provided in the next Chapter.

#Step	Agent 1			Agent 2		
	Config 1	Action 1	#Msg 1	Config 2	Action 2	#Msg 2
0	$\{C_1(a), C_2(a) -, -\}$	-	0	$\{C_4(a) -\}$	-	0
1	$\{C_1(a), C_2(a) C_3(a), -\}$	Infer	0	$\{C_4(a) Ask(2, 1, C_3(a))\}$	Infer	1
2	$\{C_1(a), C_2(a) C_3(a), Ask(2, 1, C_3(a))\}$	Comm	1	$\{C_4(a) Ask(2, 1, C_3(a))\}$	Idle	1
3	$\{C_1(a), C_2(a) C_3(a), Tell(1, 2, C_3(a))\}$	Infer	2	$\{C_4(a) Ask(2, 1, C_3(a))\}$	Idle	1
4	$\{C_1(a), C_2(a) C_3(a), Tell(1, 2, C_3(a))\}$	Idle	2	$\{C_4(a) Tell(1, 2, C_3(a))\}$	Comm	2
5	$\{C_1(a), C_2(a) C_3(a), Tell(1, 2, C_3(a))\}$	Idle	2	$\{C_4(a) C_3(a)\}$	Infer	2
6	$\{C_1(a), C_2(a) C_3(a), Tell(1, 2, C_3(a))\}$	Idle	2	$\{C_4(a) C_5(a)\}$	Infer	2

Table 4.2: One possible run of the system

4.5 Algorithm Design

In order to have an efficient RBS on small devices, we need to take into account particularly the memory consumption, communication mechanism, rule-base size along with the rest of the components. Contrary to the other algorithms, we do not store any states of conditions. Only variables and their values are stored in a key-value pair whenever a variable and its value is found. Thus occupying space only when a variable needs binding. Once a variable is bound to a value, it can be re-used in the future for the same variable. In order to run a system on a small device, the rule-base has to be small in size. Reducing rules can affect the accuracy of a system. Our novel approach towards reducing the rules in a rule-base is based on the preferences provided by the end user. This method only processes a

subset of rules that are required for a particular scenario. As an example, a user who is in the office does not need a rule which has to deal with his/her home. Processing the home-based rules burden the whole system. Instead, we do not consider them unless required. The proposed preference model and its complete working mechanism with case studies are provided in the next Chapter. The structure of rules ordering/priority can be opt-in as an added optimization feature. Our rule matching mechanism to create a conflict set checks the predicate first, if a predicate match is found in the working memory then it further checks the rule condition, otherwise discards the rule without further proceedings. A flag is set to monitor each match, and if flag value is 1 for a given condition in a rule, then it proceeds to the next condition. Whenever 0 is encountered, it represents that the rule can not fully match with the facts and the process is terminated, and the next rule is selected for a check. The working memory is fixed so a user can provide a convenient amount of memory that can be spared. The devices can trigger communication when rules specifically require to communicate. Furthermore, the preferences, as discussed in the next Chapter, reduce the number of processable rules to the least possible number without affecting the outcome of the system.

4.5.1 Match: Conflict Set Generation

The rule matching or conflict set generation algorithm generates a set of applicable rule instances according to current contexts or working memory

facts. That is, for each rule, the algorithm matches all its antecedents to the facts from the working memory, if all antecedents of a rule are matched then it will check if the consequent is already in the working memory or not, if not then the corresponding rule instance will be added to the conflict set. This process will be repeated until no more rule matches. The conflict set may contain more than one rule instance with different priorities, as well as the same rule, may have multiple instances. In order to understand the algorithm, we discuss here some of the key terms involved.

- The Rule-base is represented by \mathbf{R} which contains the set of rules of an agent.
- The working memory of an agent is represented by \mathbf{WM} , which contains the set of facts or current contexts.
- \mathbf{R}_s : a single complete rule that has its LHS and RHS or condition and consequent part.
- \mathbf{R}_b : rule body which only holds the LHS side of a rule.
- \mathbf{RHS} or consequent part of the rule is represented by \mathbf{R}_c
- \mathbf{R}_{ap} : rule atom predicate holds the value of the predicate only e.g., *Person* in *Person(Alan)*
- \mathbf{R}_{at} : holds the rest part e.g *Alan* in *Person(Alan)*
- \mathbf{F}_c : current fact being in process or selected for matching.

- F_{cp} and F_{ct} are used in same analogy as discussed above for R_{ap} and R_{at} with the only difference that they reside in the working memory.
- P_{ra} : patterns in rule body.
- **VAR**: arraylist to hold KEY and VALUE.

The graphical representation of the matching algorithm with different components is depicted in Figure 4.2. In this Figure, we have a rule-base, from which a single rule is chosen for the matching purpose. The body of the rule is taken and further divided into its atomic parts. The predicate of the atom is separated from its term, and the same procedure is carried out for the facts in the working memory. Figure 4.2 and Figure 4.3 is deliberately made simple for understanding purpose, as there are certain checks which are performed during execution. These checks decide if the following condition should be matched or not.

In the Figure 4.3, we have two different scenarios. On the left-hand side, we have a condition within a rule which has a variable in it. The algorithm first matches the predicate part. Once the predicate part matches with any of the fact's predicate, then it will proceed to the next step and perform different checks. In this case, we have a variable ?x which can hold any value and here its value is Alan. The identifier for the variable, e.g. ?x is stored in the KEY. The KEY then stores the VALUE as Alan. On the Right Hand Side (RHS), we have a comparison of constants. Within the condition, instead of variable this time, we have a constant. Since the predicate matches, it

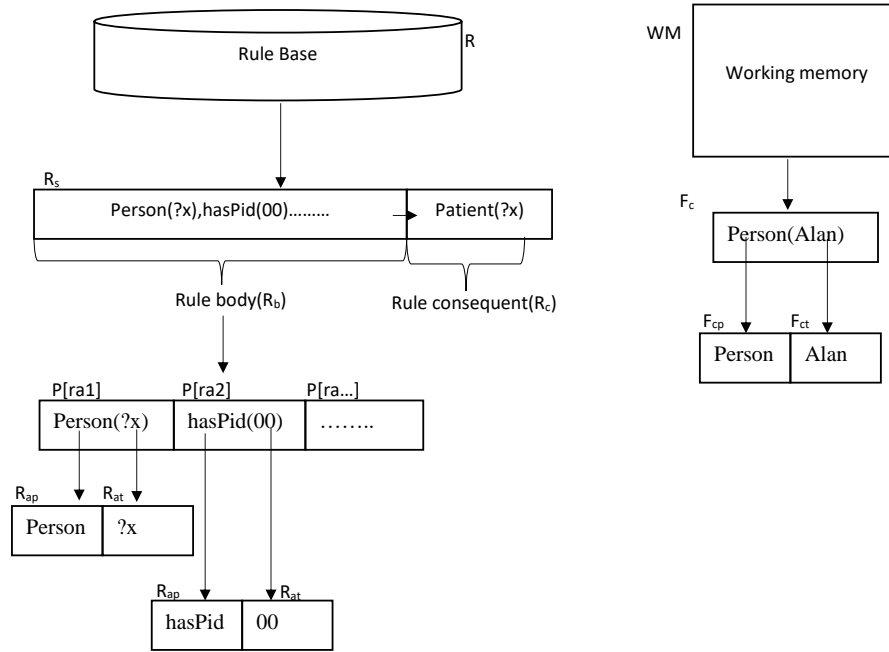


Figure 4.2: Step by step rule and fact processing

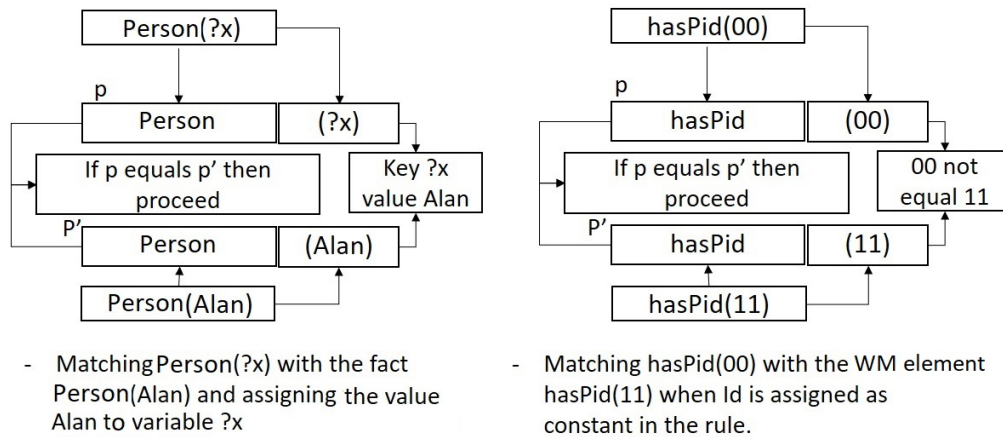


Figure 4.3: Different matching scenarios of the proposed algorithm

checks that there is no variable in the rule and the only value available is constant, which does not match with the fact-value, i.e. 11. Hence the rule is discarded without any further processing. The algorithm can process both the variable and constant in the same condition of a rule. The Algorithm 1 describes the steps involved in conflict set generation.

4.5.2 Select: Conflict Resolution

In this phase, the conflict between rule instances residing in the conflict set is resolved. Conflict resolution is the order that a rule instance is removed from the agenda or conflict set and its actions executed. In this implementation, we only use *rule ordering strategy* using the rule priority, which is an integer, determines which rule should be executed before the others. The Algorithm 2 describes the steps involved in conflict resolution, which is selecting one rule instance from the conflict set that has the highest priority. If there are multiple rule instances with the same priority exist, the rule instance to be executed is selected randomly.

4.5.3 Act: Execution Of The Selected Rule Instance

Execution of a rule instance is straight forward. When the rule instance selected from the conflict set is forwarded for execution, its consequent is added to the working memory as well as processed for further actions depending on the nature of the rule. Consequent of the rule instance can be in the form of

Input: \mathbf{R} : Rule-Base, \mathbf{WM} : Working Memory

[\mathbf{R}_s : A single rule, \mathbf{R}_i : A rule instance, \mathbf{R}_b : Rule body, \mathbf{R}_c : Rule consequent, \mathbf{R}_a : Rule atoms in the body, \mathbf{R}_{ap} : Rule atom predicate, \mathbf{R}_{at} : Rule atom terms, \mathbf{F}_c : Current fact, \mathbf{F}_{cp} : Current fact predicate, \mathbf{F}_{ct} : Current fact terms, \mathbf{P}_{ra} : Patterns in rule body, \mathbf{VAR} : Arraylist to hold KEY and VALUE]

Result: CS: Conflict set

```

1  START
2  for  $r = 0$  to size of  $\mathbf{R}$  do
3      Clear  $\mathbf{VAR}$ 
4       $\mathbf{R}_s = \mathbf{R}[r]$ 
5      Find patterns in  $\mathbf{R}_b$  of  $\mathbf{R}_s$ 
6      Add to Array  $\mathbf{P}_{ra}$ 
7      Flag: Array of size equal to  $|\mathbf{P}_{ra}|$ 
8      for  $pra = 0$  to size of  $\mathbf{P}_{ra}$  do
9          Select  $\mathbf{P}_{ra}[pra]$ 
10         Separate  $\mathbf{R}_{ap}$  from  $\mathbf{R}_{at}$ 
11         for  $f=0$  to size of  $\mathbf{WM}$  do
12              $\mathbf{F}_c = \mathbf{WM}[f]$ 
13             Separate  $\mathbf{F}_{cp}$  from  $\mathbf{F}_{ct}$ 
14             if  $\mathbf{R}_{ap} == \mathbf{F}_{cp}$  then
15                 if  $\mathbf{R}_{at} == \mathbf{F}_{ct}$  || pattern( $\mathbf{R}_{at} == \mathbf{F}_{ct}$ ) then
16                     Add 1 to Flag
17                      $\mathbf{KEY} = \mathbf{R}_{at}$ 
18                      $\mathbf{VALUE} = \mathbf{F}_{ct}$ 
19                     if ( $\mathbf{VAR}$  does not contain  $\mathbf{KEY}$ ) then
20                         Add  $\mathbf{KEY}$  to  $\mathbf{VAR}$ 
21                         Add  $\mathbf{VALUE}$  to  $\mathbf{VAR}$ 
22                     end
23                 end
24             else
25                 Add 0 to Flag exit nested loop
26             end
27         end
28     end
29     end
30     if Flag does not contain 0 then
31         for  $var = 0$  to size of  $\mathbf{VAR}$  do
32              $\mathbf{Key} = \mathbf{VAR}[var]$ 
33              $\mathbf{Value} = \mathbf{VAR}[var+1]$ 
34              $\mathbf{R}_i =$  Replace  $\mathbf{Key}$  with  $\mathbf{Value}$  in  $\mathbf{R}_s$ 
35              $\mathbf{R}_c =$  consequent( $\mathbf{R}_i$ )
36              $var \leftarrow var+2$ 
37         end
38         if  $\mathbf{WM}$  does not contain  $\mathbf{R}_c$  &&  $\mathbf{CS}$  does not contain  $\mathbf{R}_i$  then
39             Add  $\mathbf{R}_i$  to CS
40         end
41     end
42 end
43 END

```

Algorithm 1: Conflict set generation

Input: CS : Conflict set [P_o : Priority Operator, SPR : Same priority rules, C_{ics} : An element of CS , R_{ip} : Rule instance priority]

Result: to_fire = A selected rule instance to be fired

```

1 START
2  $P_o = 0$ 
3 for  $cs = 0$  to size of  $CS$  do
4    $C_{ics} = CS[cs]$ 
5   get  $R_{ip}$  for  $C_{ics}$ 
6   if  $R_{ip} > P_o$  then
7      $P_o = R_{ip}$ 
8      $to\_fire = C_{ics}$ 
9   end
10  else if  $P_o == R_{ip}$  then
11    Add  $C_{ics}$  to  $SPR$ 
12  end
13 end
14 if  $|SPR| > 0$  then
15   Add  $to\_fire$  to  $SPR$ 
16    $to\_fire =$  select a random instance from  $SPR$ 
17 end
18 END

```

Algorithm 2: Conflict resolution

communication directive, which may invoke the communication process or it can be a fact as a newly derived context to be added to the working memory or taking any other action. In order to achieve this, as we have already mentioned, the flag indicates a rule nature. If the flag is ‘G’, then a goal has been achieved, consequent will be added to the working memory, and the system needs to halt. Similarly, the flag ‘C’ indicates that the communication part needs to be invoked for this specific execution of rule instance. On the other hand, the flag ‘D’ indicates that the consequent will only be added to the working memory.

4.5.4 Working Memory Updating

The working memory of an agent carries facts which can be initial facts, the newly inferred facts as a result of the execution of any rule, or the communicated facts received as messages from other agents. In any case, it provides a holder for the available current contexts and to perform context reasoning. In the whole system design and implementation processes where the emphasis is given on the resource constraints, memory is one of the key resources we aim to save. The limit on the size of the working memory is to ensure it does not exceed the maximum number of contexts it can store at any given time, but the facts are generated at almost every iteration and keeping the facts that are more vital to the execution is a crucial task. In our implementation, the working memory is a fixed size array. The working memory of an agent is divided into static memory and dynamic memory. The dynamic

memory is bounded in size, where one unit of memory corresponds to the ability to store an arbitrary context. The static part contains initial facts to start up the system; thus, the size is determined by the number of initial facts. The dynamic part contains newly derived facts as the agent performs context-aware reasoning. Only facts stored in the dynamic memory may get overwritten, and this happens if an agent's memory is full or a contradictory context arrives in the working 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 dynamic memory is full. Because of the limited dynamic memory, there might be the case when the system can go into an infinite execution if there is no forceful stop, and the goal is not achievable. To overcome this issue we set the number of iteration equal to the number of rules we have to ensure that every rule is checked and in case of no matches is found, instead of abrupt behaviour it will halt itself, saving resources of the host system. The Algorithm 3 describes the steps involved in the execution of the selected rule instance and the updating of the working memory.

4.5.5 Communication And Subroutine Handling

Besides the conventional rule firing and updating the working memory facts, the application is also capable of handling different behaviour, which are

Input: **to_fire**: A selected rule instance to be fired [**R_c**: A communication rule instance, **R_g**: A rule instance contains a goal context, **R_d**: A deduction rule instance, **R_f**: Rule Flag, **R_{cons}**: Consequent, **MAX_SIZE**: memory size]

Output: Rule instance executed, consequent added to **WM** and corresponding action performed.

```

1 START
2 to_fire from conflict resolution and Rcons is the consequent
3 if Rg then
4   if Rcons is a conflicting context then
5     | Overwrite the contradictory context with Rcons
6   end
7   else if  $|WM| < MAX\_SIZE$  then
8     | Add Rcons to WM
9   end
10  else
11    | Overwrite an existing context with Rcons
12  end
13  Goal Reached
14  Execution Halts
15 end
16 else
17   if Rcons is a conflicting context then
18     | Overwrite the contradictory context with Rcons
19     if Rc then
20       | initiate communication module
21     end
22   end
23   else if  $|WM| < MAX\_SIZE$  then
24     | Add Rcons to WM
25     if Rc then
26       | initiate communication module
27     end
28   end
29   else
30     | Overwrite an existing context with Rcons
31     if Rc then
32       | initiate communication module
33     end
34   end
35 end
36 END

```

the outcomes of the consequent of a rule instance. For instance, agents can exchange messages regarding their current contexts. In order to achieve this, an agent has to invoke communication subroutine. Where the communication subroutine is responsible for exchanging the information from one device to another. In [109], *Ask* and *Tell* primitives have been defined to achieve communication between agents (e.g., two smart devices), *Ask* is used when one device asks for some contextual information, similarly *Tell* is used to answer the ask or simply conveying some contextual information without being asked. However, in practice, how the contextual information is sent or received is a matter of question. In our implementation, the devices can communicate via SMS and Bluetooth. We further proposed that in order to achieve efficient communication, a table has to be maintained and distributed among all the connected devices. This table contains a list of available communication modes supported in the domain. Each device is assigned with a numeric ID, and this ID can be used in the $Ask(i, j, p(t_1, t_2))$ and $Tell(i, j, p(t_1, t_2))$, which will also keep the logical structure of the rule intact. The i and j specify the *FROM* and *TO* respectively. If we assign them numbers, it can specify which devices are communicating with each other. For example, when $i = 2$ and $j = 3$, the *Ask* primitive becomes $Ask(2, 3, p(t_1, t_2))$, where $p(t_1, t_2)$ is an atomic context which neither contains an *Ask* nor a *Tell*, and according to the Table 4.3 where the ID 3 is associated with a caregiver device (as agent 3) and 2 is associated with a patient care device (as agent 2). In case if the patient care agent wants to communicate with

the blood pressure monitor agent, it can use the same format by specifying the ID of the blood pressure monitor device. The rest of the columns specify the different available modes of communication and their respective addresses. In the case of Bluetooth communication, these devices have to be paired with each other. Once paired names are added to a pair list, they can be specified in the table in order to initiate communication. Once the agent's IDs are specified, the communication mode can be specified explicitly by adding the communication mode at the beginning of the *Ask* and *Tell* rule, e.g., *Bluetooth(Ask(i, j, p(t₁, t₂)))*, which will be taken as agent *i* wants to communicate with agent *j* using Bluetooth only. In case if no pre-rule communication method is defined, then any of the available communication modes can be used. While this is so far handling communication using the rules, but in order to make it work every communication method has to be attached with its respective handler and a method has to be specified which can understand the rule and interpret it into Android specific format. These rules before triggering will be checked with the *Ask* and *Tell* rules. If any of them is found, a subroutine will be called to handle the rule, which will extract its FROM and TO from the agent ID table along with the communication addresses and act accordingly. The communicated contexts, when received by a receiver agent, are stored in a buffer before putting them into the working memory. If the receiving agent is in the middle of the execution, it will first complete its current execution, and in the next iteration, it will add the received contexts from the buffer to the working memory and will

Agent ID	Bluetooth	IP address	ICCID (Cell number)
1	BP monitor	x.y.z.a	111222333
2	Patient care	x.y.z.b	111222444
3	Caregiver	x.y.z.c	111222555

Table 4.3: Agent ID table

continue further processing.

4.5.6 Time And Space Complexity Of Core Algorithms

In this section we try to find the asymptotic complexity of our proposed algorithm both in terms of the time and space. As discussed earlier (See 3.5.1 that the time and space complexity of the RETE algorithm along with other eager evaluation algorithm can reach to the $O(WM^{RC})$ [71, 99].

Match Algorithm Complexity

In the Table 4.4, we found the time complexity for Algorithm 1.

Calculating the (cost * frequency) for each step to find the dominating factor of the algorithm in order to find the order of growth in terms of n. It is noted that the asymptotic time complexity of the algorithm is $O(n^2)$

$$\text{Let } T(n) = c_1 + nc_1 + c_2n + c_3n + c_4n + c_5n + c_6n + c_7n + c_7n^2 + c_8n^2 + c_9n^2 + c_{10}mn^2 + c_{11}mn^2 + c_{12}mn^2 + c_{13}mn^2 + c_{14}mn^2 + c_{15}mn^2 + c_{16}mn^2 + c_{17}mn^2 + c_{18}mn^2 + c_{19}mn^2 + c_{20}mn^2 + c_{21}mn^2 + c_{22}mn^2 + c_{23}n + c_{24}n^2 + c_{24}n + c_{24}n^2 + c_{25}n^2 + c_{26}n^2 + c_{27}n^2 + c_{28}n^2 + c_{29}n^2 + c_{30}n^2 + c_{31}n^2 + c_{32}n^2$$

Algorithm	<i>cost</i>	<i>frequency</i>
For r=0 to size of R	c ₁	n+1
Clear VAR	c ₂	n
R_s =R[r]	c ₃	n
Find patterns in R _b of R _s	c ₄	n
Add to Array P _{ra}	c ₅	n
Flag : Array of size equal to P _{ra}	c ₆	n
For ra = 0 to size of R_a do	c ₇	n(n+1)
Select R_a [ra]	c ₈	n ²
Seperate R _{ap} from R _{at}	c ₉	n ²
For f=0 to size of WM do	c ₁₀	n ² m
F_c = WM [f]	c ₁₁	n ² m
Seperate F_{cp} from F_{ct}	c ₁₂	n ² m
if R_{ap} == F_{cp} then	c ₁₃	n ² m
if R_{at} == F_{ct} pattern(R_{at} == F_{ct}) then	c ₁₄ + c ₁₅	n ² m
Add 1 to flag	c ₁₆	n ² m
KEY = R _{at}	c ₁₇	n ² m
VALUE = F_{ct}	c ₁₈	n ² m
if (VAR does not contain KEY) then	c ₁₉	n ² m
Add KEY to VAR	c ₂₀	n ² m
Add VALUE to VAR	c ₂₁	n ² m
else Add 0 to Flag and Exit Loop	c ₂₂	n ² m
if Flag does not contain 0 then	c ₂₃	n
For var=0 to size of VAR do	c ₂₄	n(n+1)
Key = VAR[var]	c ₂₅	n ²
Value = VAR[var + 1]	c ₂₆	n ²
R_i Replace Key with Value in R_s	c ₂₇	n ²
R_c = consequent(R_i)	c ₂₈	n ²
var ← var+2	c ₂₉	n ²
if WM !contain R _c AND CS !contain R _i then	c ₃₀ + c ₃₁	n
Add R _i to CS	c ₃₂	n

Table 4.4: Conflict set generation Algorithm Complexity

$$= c_1 + n(c_1 + c_2 + c_3 + c_4 + c_5 + c_6 + c_7 + c_{23} + c_{24} + c_{30} + c_{31} + c_{32}) + m(c_{10} + c_{11} + c_{12} + c_{13} + c_{14} + c_{15} + c_{16} + c_{17} + c_{18} + c_{19} + c_{20} + c_{21} + c_{22}))$$

$$\text{let } c_{33} = c_1 + c_2 + c_3 + c_4 + c_5 + c_6 + c_7 + c_{23} + c_{24} + c_{30} + c_{31} + c_{32}$$

$$\text{let } c_{34} = c_8 + c_9 + c_{25} + c_{26} + c_{27} + c_{28} + c_{29}$$

$$\text{let } c_{35} = c_{10} + c_{11} + c_{12} + c_{13} + c_{14} + c_{15} + c_{16} + c_{17} + c_{18} + c_{19} + c_{20} + c_{21} + c_{22}$$

The equation becomes

$$c_1 + c_{33}n + (c_{34} + mc_{35})n^2$$

$$\text{let } c_{36} = c_{34} + c_{35}m - m \text{ being constant}$$

$$= c_1 + c_{33}n + (c_{36})n^2$$

So the $T(n)$ becomes

$$T(n) = c_1 + c_{33}n + (c_{36})n^2$$

We remove the $c_1 + c_{33}n$ to get $(c_{36})n^2$ only, which is a dominant factor of this algorithm. We can further remove the constant factor to get the order of n here.

The space complexity of our algorithm is $O(n)$ since the two main arrays which carry the patterns and flag has the highest complexity of $O(n)$. Comparing the time and space complexity of the proposed algorithm, we have an advantage on the space side. Also, in the algorithm, there is always a trade-off between space and time. Time can be accommodated by providing more space and vice versa. However, the worst cases of our algorithm are

Algorithm	<i>cost</i>	<i>frequency</i>
$P_o = 0$	c	1
For $cs = 0$ to size of CS	c_1	n
$C_{ics} = \mathbf{CS}[cs]$	c_2	n
get R_{ip} from C_{ics}	c_3	n
if $R_{ip} > P_o$	c_4	n
$P_o = R_{ip}$	c_5	n
$to_fire = C_{ics}$	c_6	n
end		
else if $P_o == R_{ip}$ then	c_7	n
Add C_{ics} to SPR	c_8	n
end		
end		
if $SPR > 0$ then	c_9	1
Add to_fire to SPR	c_{10}	1
$to_fire = \text{random}(SPR)$	c_{11}	1
end		
END		

Table 4.5: Algorithm Conflict Resolution Running Cost

considerably low and also usable on any low resource devices efficiently. As the framework is intended for use on small devices and the smart systems are usually designed with a small set of rules.

Conflict Resolution Algorithm Complexity

The conflict resolution input depends on the number of rules that are selected in the conflict set. It iterates through the CS and finds the highest priority for execution. The time complexity and space complexity is provided in Table 4.5 for Algorithm 2

In this algorithm, let T represent the time it takes for the algorithm to

execute. $T = c + (c_1 \dots c_8)n + c_9 + c_{10} + c_{11}$

adding $C = c + c_9 + c_{10} + c_{11}$ and $M = c_1 + \dots + c_8$

Such that, $T = C + Mn$ considering the most significant term we get the $O(n)$ complexity, this goes same for the space complexity as there is only one array that holds the values for CS.

Execution Of Rule Complexity

The rule execution is quite straight forward. When a rule is passed by the conflict resolution phase, then it is ready to be fired. The fired rule can have different impacts i.e it can add something to the working memory or initiate communication as in case of *ask,tell* rules or simple reach the goal and terminate the process. The Table 4.6 provides the complexity for Algorithm 3.

In the rule execution complexity the calculation is very clear, as the frequency of all the iterations are static. In order to calculate the time let us assume T is time required to execute the rule execution.

$T = (1 * C_1) + \dots + (1 * C_{19})$ yielding $T = 1(C_1 + \dots + C_{19})$

Considering only the most significant part for the *complexity* we have $O(1)$.

In terms of space, the algorithm only read from the memory which is already calculated in the previous algorithms and creating no new space to be added.

Algorithm	<i>cost</i>	<i>frequency</i>
START		
If R_g then	C_1	1
If R_{cons} is a conflicting context Then	C_2	1
Overwrite the contradictory context with R_{cons}	C_3	1
Else If $ WM < MAX_SIZE$ then	C_4	1
Add R_{cons} to WM	C_5	1
Else Overwrite an existing context with R_{cons}	C_6	1
Goal Reached	C_7	1
Execution Halts	C_8	1
Else		
If R_{cons} is a conflicting context	C_9	1
Overwrite the contradictory context with R_{cons}	C_{10}	1
If R_c then	C_{11}	1
initiate communication module	C_{12}	1
Else If $ WM < MAX_SIZE$	C_{13}	1
Add R_{cons} to WM	C_{14}	1
If R_c then	C_{15}	1
initiate communication module	C_{16}	1
Else Overwrite an existing context with R_{cons}	C_{17}	1
If R_c then	C_{18}	1
initiate communication module	C_{19}	1
END		

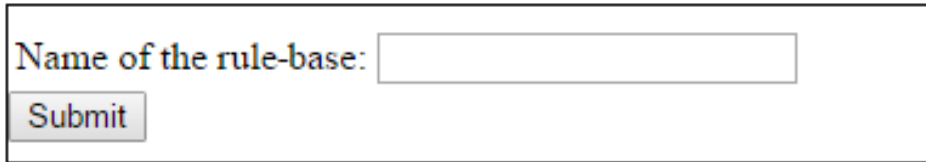
Table 4.6: Cost for Executing a Selected Rule Instance

4.6 Rule Generation And System Design

Rules are the central driving unit behind the whole framework. Rules give the system the ability to think. In this model, two different ways to create a rule is provided. One is the web-based rule formation tool, while the other one is an enhanced prototype of Onto-HCR translator, which integrates as a plugin in the protégé. The section following describe each method in detail.

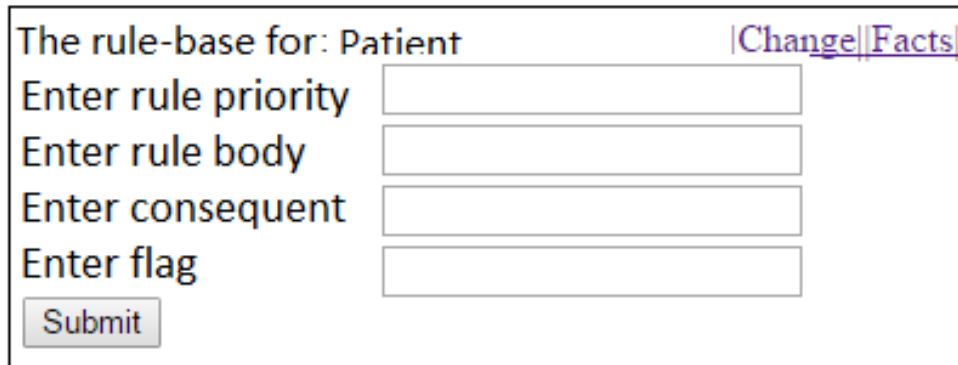
4.6.1 Web Based Rule Generator

The desktop interface is a web-based application, which uses Apache web server, MySQL Database, JQuery and PHP language as its main components besides HTML for the interface design. The application is platform independent and runs in any standard browser on different platforms, e.g. Windows, Linux, Macintosh with minimum setup. This interface can be made online to be accessed from any computer with an Internet connection. The user can add rules from this interface, and it allows validating both the Left Hand Side(LHS) or body and Right Hand Side(RHS) or head (or consequent) of any rule provided. If the rule qualifies the format specified, it will be returned as valid; otherwise, the user will be prompted to enter a rule according to the intended format. The interface allows creating an agent program by receiving a set of rules and initial working memory facts. The various phases of the desktop interface are provided in Figures 4.4, 4.5, and 4.6. In Figure 4.4 (a), a system developer can create a rule-base. Once a rule-base is created,



Name of the rule-base:

(a) Creating an agent's rule-base



The rule-base for: Patient [|Change|Facts|](#)

Enter rule priority

Enter rule body

Enter consequent

Enter flag

(b) Rules insertion interface

Figure 4.4: Rule-base initialization

the next interface is shown in Figure 4.4(b), is ready to receive input a set of rules. Figure 4.5 shows the validation of the rule and allowing the system developer to save it if the rule entered is according to the correct format. Figure 4.6 shows the phase where the system developer wants to enter the facts; it auto-suggest the rules as a developer starts typing so that the chances for adding irrelevant or erroneous facts are minimized. The Desktop interface produces its output in the form of a JSON ⁴ file. JSON is a lightweight data interchange format. The JSON file is then further provided as an input to the Android application.

⁴JSON-<http://www.json.org/>

The rule-base for: Patient [Change](#) [Facts](#)

Enter rule priority

Enter rule body

Enter consequent

Enter flag

Rule priority specified: 1
Rule body specified: Person(?p), hasPatientID(?p,?pid), PatientID(?pid)
Consequent specified: Patient(?p)
Flag specified: D

Rules format validation
Your entered rule body Person(?p), hasPatientID(?p,?pid), PatientID(?pid) is correct.

Consequent format validation
Your entered consequent Patient(?p) is correct.

Do you wish to save the rule?

Figure 4.5: Validation of Horn-clause rules

Enter facts for **patient**

The auto suggest recommends rules as you insert

Patient

Person(?p), hasPatientID(?p, ?pid), PatientID(?pid) -> Patient(?p) ▼

Figure 4.6: Facts interface with auto suggestion

4.6.2 Onto-HCR Protégé Plugin For Heterogeneous Ontologies

To extract the rules from different ontologies, a plugin based on our previous work [110] is developed for Protégé IDE. It allows the user to select OWL 2 RL ontology files augmented with SWRL rules and translate into a set of plain text Horn clause rules, once translated the user can further edit the rules, e.g., to inset priorities, flags or preferences. Once the editing is complete, the translated Horn clause rules could be used to model the desired context-aware system.

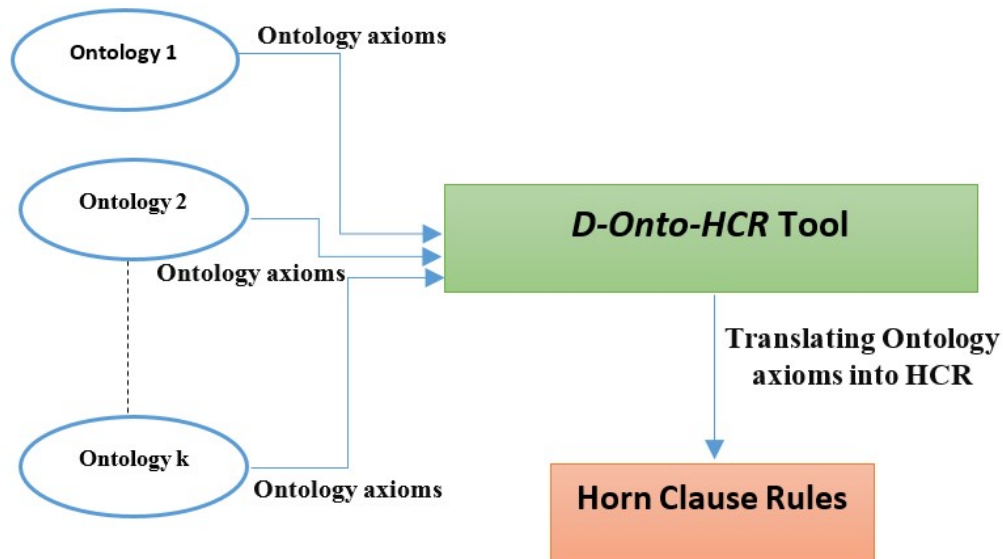


Figure 4.7: Distributed semantic knowledge translation process

Overall the plugin is a few step process. First, it let the user select the downloaded ontologies which are in OWL/XML format. The plugin then

translates it into plain horn-clause rules; once the rules are generated, the user can select another file in order to get rules from multiple ontologies. Finally, the user selects the generated file for editing, and once the edits are done, the resulting file can be used to create any semantic web rule-based application. Figures 4.8, 4.9 and 4.10 show the various steps involved in translating the ontologies. This step can be repeated for different ontologies individually, which will be translated into a single file containing the plain text Horn-clause rules extracted from multiple ontologies. The following subsections describe the relevant details of various aspects of the plugin.

Development Environment

The development of Protégé plug-in is carried out in the Eclipse IDE for Protégé version 3.4.1. In order to develop a plugin, there are quite a few steps involved. First, set up the project and name it after the plugin as DontoHCR. Once a project is set up, there is a need to add all the external libraries that are required by the plugin, by adding external JAR's within the IDE to the project. To create a tab widget, one has to extend the `AbstractTabWidget` class from the `protege.jar` to implement the `initialize()` method. This method runs when the plugin starts on the Protégé the main interface. This gives the user a place where user can put their code that can appear in the Protégé as a tabbed plugin. However, in order to run it on Protégé first there is the need to set the manifest file, which makes the Protégé to recognize the new plugin. For testing, change the run configuration and

set the working directory besides some other changes, to the Protégé main directory. This way when the IDE tries to run the program, it launches the Protégé interface which can run the plugin and check its functionalities. The technical components of the plugin are described below.

Working Mechanism Of The Plugin

The plugin is an OWL-API based translator, to extract rules from different ontologies. It is the high-level Java-based API that supports the creation of OWL ontologies and also enables us to manipulate the ontologies. The OWL API enables third-party developers, to create and/or customize different implementations for their components. It is helpful in loading, saving, parsing and serializing ontologies in different syntaxes such as, OWL/XML, RDF/XML, functional syntax, Manchester syntax, KRSS and Turtle syntax which are commonly used. Furthermore, it has a set of interfaces for probing, manipulating and reasoning with OWL ontologies. Some of the main features of OWL API are an axiom-centric abstraction, reasoner interfaces, validations for different OWL 2 profiles and first class change support. In terms of functionalities, the plugin input is the ontology. Which translates the set of axioms into Horn-clause rule. The set of axioms can be OWL 2RL and SWRL form. The plugin translates DL-safe rules axioms into Horn-clause rules. In addition to that, it extracts concepts from multiple ontologies and maps them correspondingly in the form of bridge rules. These bridge rules are first converted into OWL 2 RL rule format and then into

Horn-Clause rules. Figure 4.7 shows the process of translation. When the plugin is loaded, an ontology file is provided as an input. OWL parser is used then to parse the ontology into OWL API objects which then extracts the set of TBox and ABox axioms. The resultant set of TBox and ABox axioms is then translated into Horn-clause rules. The process is repeated for any number of ontologies, and the final output is a single file which contains a set of plain text Horn-clause rules.

4.7 Discussion

Based on the understanding from previous chapters, in this Chapter, we have formally proposed a framework and define its working mechanism in detail. The proposed framework platform is selected to be on Android devices. Android devices are not limited to smartphones only; rather, it can be found on small chips, TV, Set-Top Boxes and similar devices. Android having a larger user base also make it favourable for selection. Since the sensors are the main hardware interface that perceives its environment, it has to be clear in working. How the sensors raw data is gathered and by which mechanism it has to be acquired. Having context in hand makes eligible for the reasoning. For the reasoning part, the Rule-based system takes into consideration the acquired context. In other words, the context drives the rule-based system in such a way that it reacts to contexts as it arrives and based on the context, it can adapt itself or act on behalf of a user. We argued that the RETE

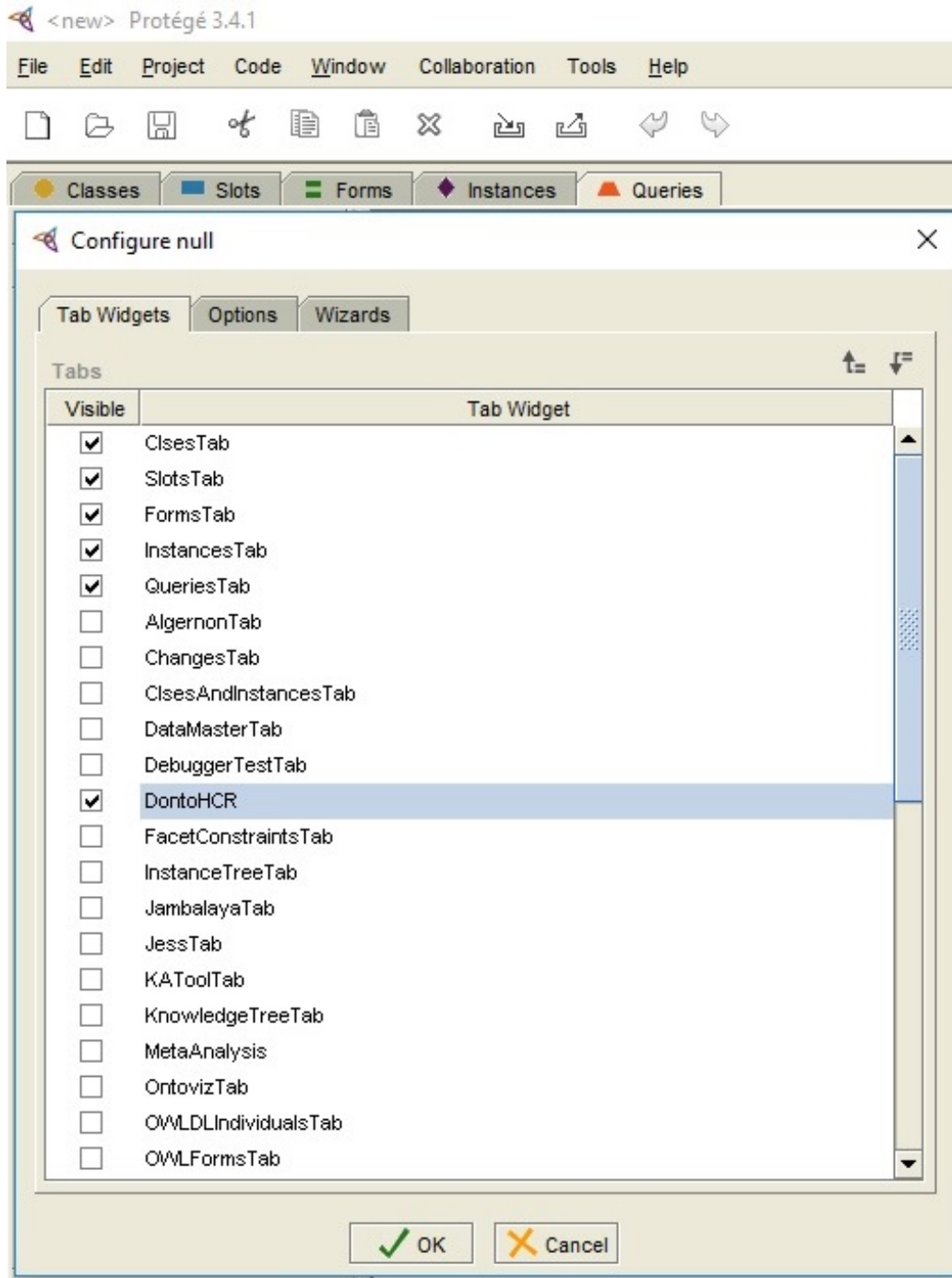


Figure 4.8: Enabling the plugin in protégé

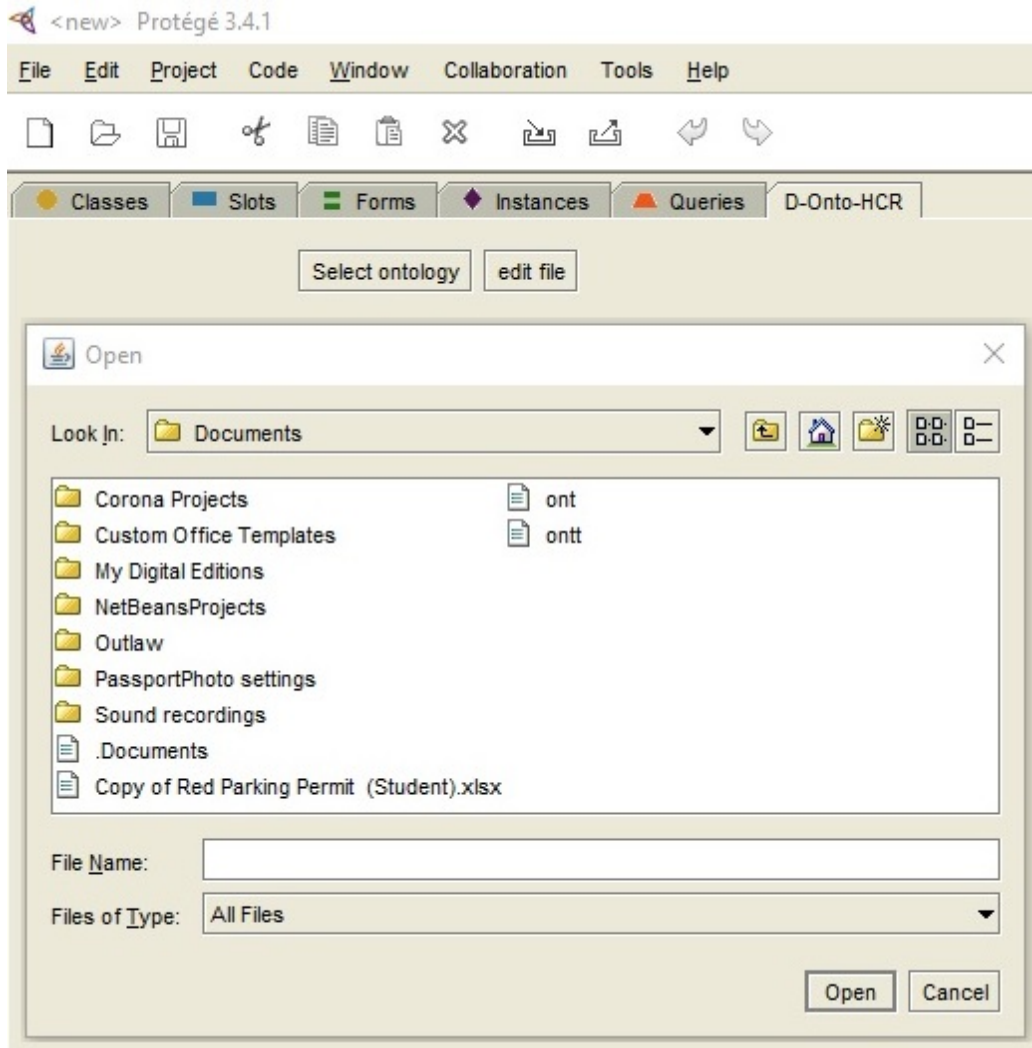


Figure 4.9: Selecting ontology for translation

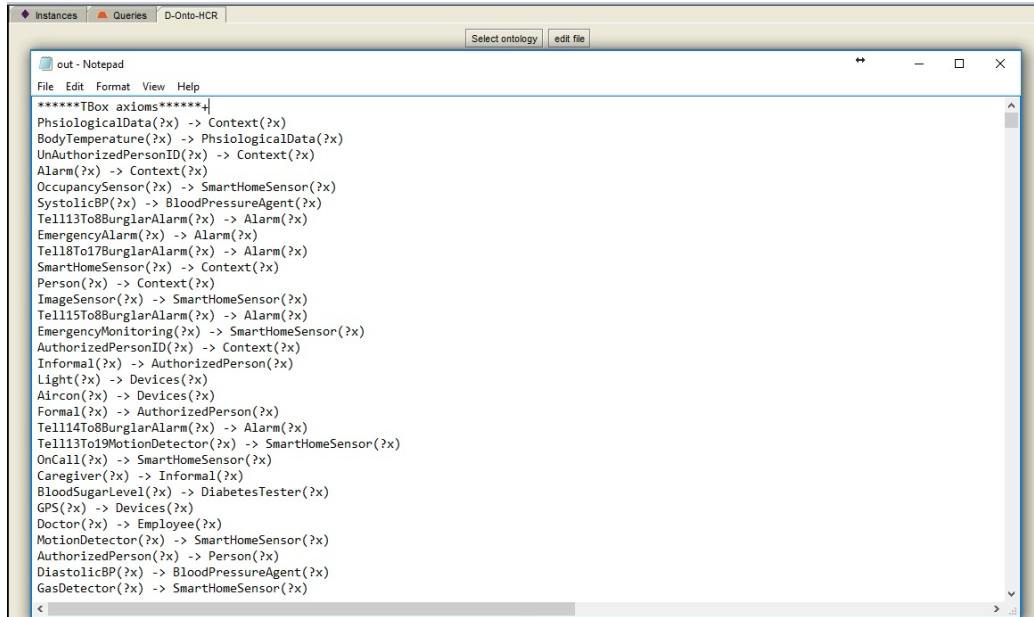


Figure 4.10: Editing the translated file

algorithm in RBS is not a recommended approach, especially when the resources are limited. Therefore we define our efficient algorithm published in [111], which consumes meagre resources, have fewer lines of code and has satisfactory time and space complexity. We also looked into the matter of rules generation, and there can be instances when we need a limited set of rules, and we do not have ontology to translate the rules. In order to solve this problem, we developed web-based rule generator which can be used to encode human expert knowledge into rules directly, which can be used on the framework rule-engine without any need of a translator. Besides rules translation from ontologies, the tool has also been tailored as a plugin to be used in the Protégé IDE as a plugin. Which can be used to translate, edit

and save different ontologies into the same file as simple Horn-clause rules. In the end, we define how to use the plugin in Protégé. Since being resource friendly algorithm is not enough, as the rules itself can upgrade/degrade the efficiency of the rule-engine, we further enhanced the rule-engine by introducing a novel approach of preferences, which not only reduce the rule-base size but also provide the user with personalized services. Preferences are defined in detail in the upcoming Chapter.

Chapter 5

Preferences

5.1 Introduction

In the previous chapter, we discussed the rule-based reasoning framework for tiny or resource-bounded devices. In this chapter, we present a preference model for personalisation of resource-bounded context-aware applications, which provides a novel approach to reduce the number of rules to be processed by the matching phase, reducing rules increase the overall efficiency of the system, aside from reduction in rules it gives the user a control to select the subsets of preferred rules. We aim to achieve this by keeping the current core of the framework untouched and do the preferences execution beforehand. As the role of preferences in multi-agent systems has received very little attention [112] we strive to define a well-structured preference selection method. Preferences main task is to provide a user with related data.

This data can be in form of recommended applications [113], personalizing notifications [114, 115], UI customization [116], rule to trigger, context-aware personalization [117] or it can provide a user with list of places a user might be interested to visit while on a tour [118]. In any method, the idea behind is to find/get the user preference and based on the preferences serve the user accordingly. As discussed in the research work, [116] the user interface of a terminal can be changed according to the preferences specified by the user. The GPII personalisation infrastructure enables auto-personalisation from preferences. When a user signs in, the automatic personalisation of signed in device starts. The login is supported by user listeners such as USB or NFC. Once logged in, the flow manager (which is mainly responsible for personalisation process) acquire the user preferences from a cloud-based preferences server. Device reporter provides information about the device the user has signed in similarly other reporters provide relevant information if any. After receiving these, the matchmaker part assembles a complete list of solutions and features that shall be configured on the device. These setting or configuration are then passed on to the lifecycle manager as instructions, to perform the actual adaptation of the device. In this setup, it may be noted that a complete cloud-based preference server is used just for managing user preferences. The preferences are also defined within the ontology, increasing the size of ontology. This method is not recommended especially in resource-bounded devices as it has to access cloud server (need active connection), use preference reasoning from ontology (need more com-

putation) and matchmaker has to define all the possible solution instead of providing some specified solutions. In [119] preferences can be predicted of a user by knowing how a user is selecting different items, e.g. applications in the current context. Defining relationships between the items selected by a user in one context is used to define latent preferences related to the user. The interest is to find those hidden reasons for which a user acts in a certain way. The similarity matrix created for user helps in identify such preferences. However, the process may face a cold start issue and needs a dedicated module to observe and predict preferences. Similarly, the approach used in [118] may also face a cold start problem in case if there are no comments for a venue or the venue is recently open. In this paper, a venue is said to be recommended to a user based on the context-aware mechanism by gathering related words related to a particular venue, e.g. Family, kids, parents all are related in a form that represents a family. If a particular venue has such words in comments, then the system process such comments in the context of users preference and recommends places accordingly. The terms are collected from Location-Based Social Networks (LBSN). Venue suggestion aims to helps users by providing personalised recommendations of places to visit, using LBSN such as Foursquare. In order to make it context-aware, it has to take the related context of the user into consideration. Such context can be the location of the user and time of the day along with others. The authors have exploited word embedding techniques to infer the vector-space representations of venues, users existing preferences, and users contextual

preferences. Another preference mining approach presented in [120] is to develop a context log for users from the user activities, then mining common context-aware preferences from these logs for different users. After mining the personal context-aware preference of each mobile user, the predicted category of contents is preferred for a given user according to the corresponding context. To put it simply, if the system infers that the user would like Puzzle games, then the recommended games will be puzzle games. However, when we keep in mind the resource limitation and to avoid the extra computation required to predict user preferences by tracking the user's current behaviour may not be an ideal solution. Instead, a user should be given control over the preferences that may be of interest to them by explicitly providing them in the knowledge base. Using this same concept, we provided an extended framework that can accommodate the preferences explicitly.

5.2 Preference In Context-Aware Agents

In this section, the extended framework by defining components that allow personalised services is discussed. In order to implement user preferences, an extra preference manager layer is added while the original working inference engine remains intact.

The main idea of user preference is to select a subset of rules based on preferences, and this allows the framework to process only the rules that are of importance to the user instead of going through all the rules. The

whole process is composed of different steps and modules which are explained throughout this chapter. The preference manager layer mainly consists of Context Set (CS), Context Monitor (CM), Preference Set Generator (PSG) while Context of Interest (COI) provided by the user beforehand is the context in which a user has interest for personalisation. Figure 5.1 shows how these components are related to each other.

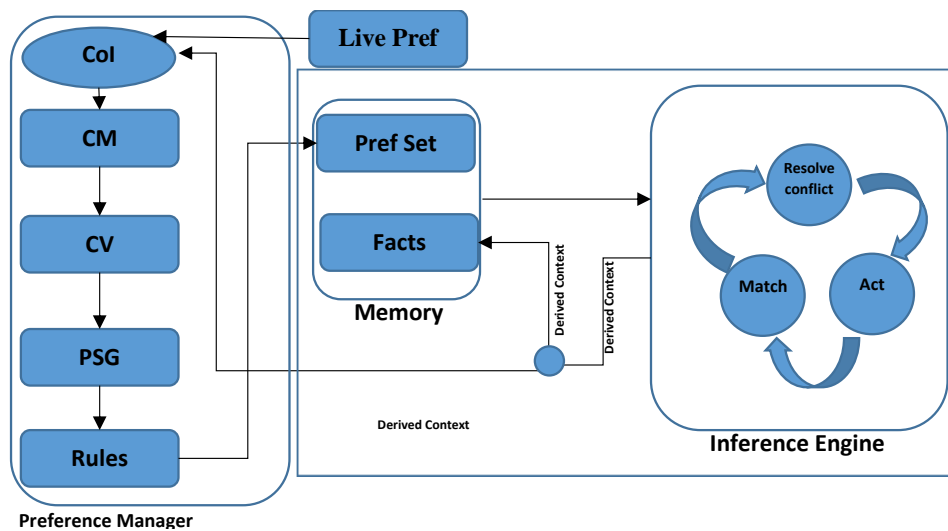


Figure 5.1: Preference generation overview

5.2.1 Context Set

The context set (CS) component is a column added to rule-base. The literals in this column against each rule works as an indicator for that particular rule. It indicates if a rule belongs to a particular context, e.g., $Person(?p), OfficeRoom(?o), hasLocation(?p, ?o) \rightarrow inOffice(?p, ?o)$ with indicator “L”

in CS can be attributed towards the context location, which represents that the rule belongs to group of rules that are part of location rules. CS may contain multiple indicators, for example, if user location and his/her blood pressure mentioned in the same rule, then CS can indicate both contexts defined with two different literals. These all CS indicators can easily indicate the contexts included in a rule. For example, a user may want preference based on location only. So the preference set will be generated where all the rules share the location indicator in the CS column. It is pertinent to mention that any rule that does not have CS indicator is a general rule, represented by “-” in the context set, and will be added to every subset that is created for preference set.

5.2.2 Context Monitor

The context monitor (CM) holds the Context of Interests (COI) of a user, i.e., it holds the values provided by the user. Context monitor after reading the values passes them to the Preference Set Generator (PSG) which defines a subset of rules based on the user preferences called preference set. This subset is then passed to the inference engine for processing. Context monitor actively monitors the contexts of interests. Any change in the context is forwarded to PSG to derive a new set of rules to be processed according to the changing preferences. The changes can be from the user or the system itself.

5.2.3 Preference Set Generator

The preference set generator (PSG) is the main part which gives the framework an ability to provide personalized services. This layer provides a subset of rules which are personalized set of rules for a current context of the user. PSG receives instructions from the CM to derive a set of personalized rules. The rule-base of an agent consists of a variety of rules; some rules may never get a chance to execute while some may be actively executed. The preference set generated at this stage has the rules which have higher chances of being fired. It replaces the main rule-base and acts as a small rule-base for the inference engine.

5.2.4 Working Mechanism

In this section, it is defined that how different components work together in order to provide personalized services to a user. All the rules are initially stored in the main knowledge base or rule-base. As the process starts, the COI is provided by the user, and the CM component retrieves the values. The CM forwards the values to the PSG. PSG further communicates with the rule-base and picks only those rules that are of interest to the user based on the values specified in the COI. The PSG makes use of the CS to fetch the desired rules. When the PSG rules are ready, these rules are provided to be used as the knowledge base for further processing. Practically addition of preference layer is the significant change, which reduces the overall burden

from main inference engine and making it more efficient in terms of reducing rules.

5.3 Type Of Preferences

Exploring the different type of contexts, and their usage yields that the preferences can be composed of at least three different types. Each of them has their selection and execution criteria. The rules can be categorized into three different categories, such as context-based, derived context-based and live context. The explanation for each of them is provided in this section.

5.3.1 Context-Based

The context-based preference is the simplest one. It makes a subset of rules, based on the user's selected context before the system starts processing. The rules are grouped by the same context set indicator or CS. A single rule can be a member of different subsets. Once the user selects the context, it can proceed to the next process by creating a subset from the main rule-base. Although it has advantages, there is one issue when a user is anticipating some context in the future to come, and it is not selected in the preference set. For that reason, the derived context based preference is used.

5.3.2 Derived Context-Based Preference

When a user is expecting some context to appear in the future and the user wants to enable the preference on that context, then it can be enabled by putting certain rules in a category which a user then keep under watch until it is derived. A good example is if a user visits the hospital for some reason other than for a check-up, then the rules associated with the person being a patient should not execute. However, if the user is visiting the hospital and his/her condition is detected as being ill, then the patient rules should apply. Figure 5.2 provides a graphical illustration of the derived context preference set generation.

5.3.3 Live Preference

Live preference comes in handy when a user wants some context to monitor continuously until it occurs. For example, if a user wants to keep logging the GPS unless a certain point comes to execute some rules. Once the system detects the context, the preference set is enabled and vice versa. A good example is a user applying some specific rules on a Sunday; in that case, the context of day is monitored unless it becomes Sunday. On other days normal rules will be applied for processing. Similarly, GPS data can be monitored to enable or disable specific rules. For example, a user only enables the office rules when the user location is detected as in the office. Once the context of office changes from office to any other, the rules will change

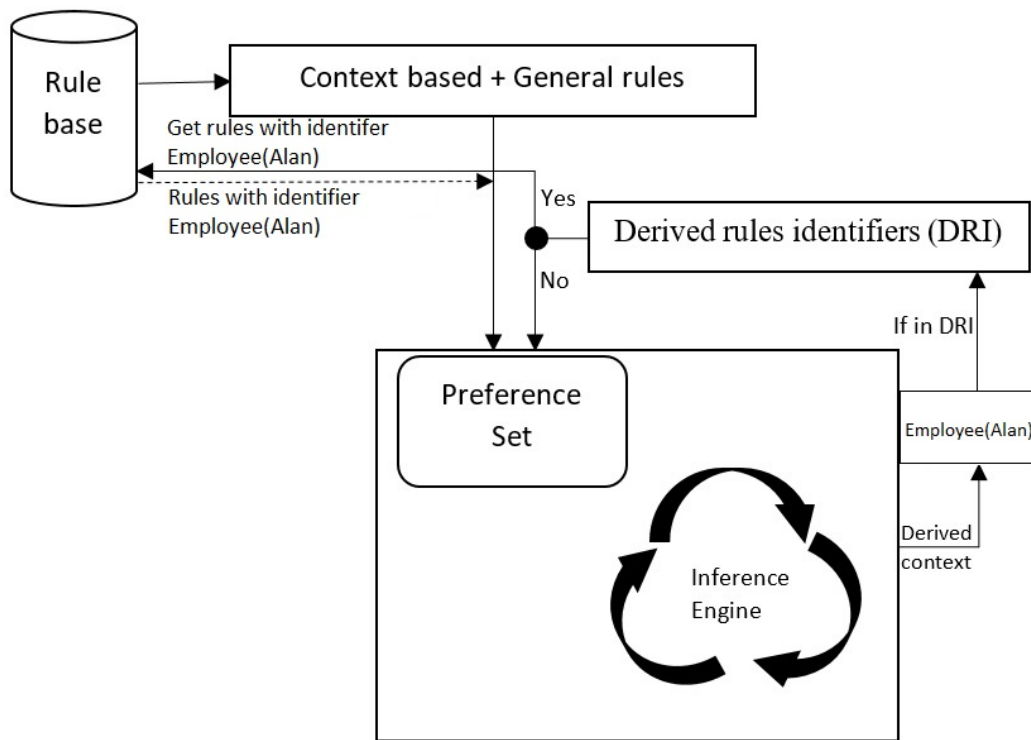


Figure 5.2: Derived context based preference set generation

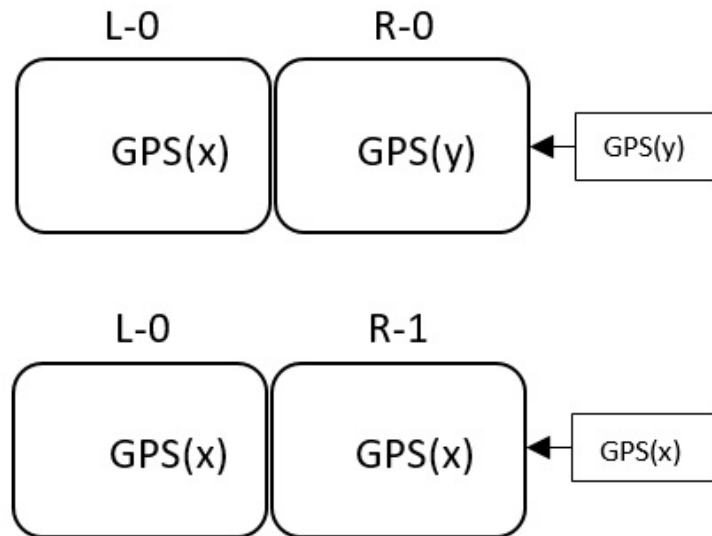


Figure 5.3: Live preference change detection

subsequently. Figure 5.3 gives a schematic view of live preference execution, while Algorithm 4 provides the stepwise essential execution of live preference.

5.3.4 Preference Set Generation

To put it simply, all the changes are directed towards the COI. COI, when sensing any change in its contents, direct a new preference set generation instruction. Based on the contents of the COI which are either context-based, derived, live or general ones. At the end of the process, all it matters is what is residing in the COI regardless of its source. The new preference set is generated and put forward for the system for execution. The Algorithm 5 shows the process.

```
Result: Live Preference Set
START
if  $Flag \neq 00$  then
  | if  $Context\ in\ L == Context\ in\ R$  then
  | | Select all from Rule-base where identifier equals context
  | | received
  | | Flag=00
  | end
else
  | Do Nothing
end
```

Algorithm 4: Live preference set generation for single context

5.4 Case Studies

In this section, different case studies are studied to check the framework integrity and results.

5.4.1 Case Study 1

Published in [110]. Some rules are used as general rules to be added into the preference set, which will be added to the preference set in any case. Due to the long set of steps from different ontologies, annotations and preference set we limit our case study to elaborate more the preferences side, and how the preference sets are generated, while the execution of the rules remains the same as described. Therefore in our case study, we take two different ontologies, one of the ontologies deal with the smart office while the second ontology is for patient care system. However we can use both the ontologies, and then use preferences in them so that a user when in the office can be cared

Input: **COI:**Context of Interest, **R:** Rules, **F_e:** Facts from external agents or sensors, **F_d:** Facts derived, **CS:** Context Set, **Regex:** regular expression

Output: Preference Set based on COI

```

1 START
2 if Regex(COI) == [a-zA-Z] then
3   | Fetching Simple preference based on literals
4   | for  $r \rightarrow [R]$  do
5   |   | if  $\exists x \in \mathbf{COI}$  such that  $x \in \mathbf{CS}[r]$  then
6   |   |   | Add  $r$  to Preference Set
7   |   | end
8   | end
9 else if Regex(COI) == [a-zA-Z] + ([a-zA-Z0-9]+) OR
10  | [a-zA-Z] + ([a-zA-Z0-9]+, [a-zA-Z0-9]) then
11  |   | Derived or Live preference of the form A(b) or B(b,c)
12  |   | for  $r \rightarrow [R]$  do
13  |   |   | if  $\exists x \in \mathbf{COI}$  such that  $x \in \mathbf{CS}[r]$  AND  $x \in \mathbf{F}_e$  then
14  |   |   |   | Add  $r$  to Preference Set
15  |   |   | end
16  |   | end
17 else if CS[r] == "-" then
18  |   | Add  $r$  to general rule
19 end

```

Algorithm 5: Preference set creation based on COI values

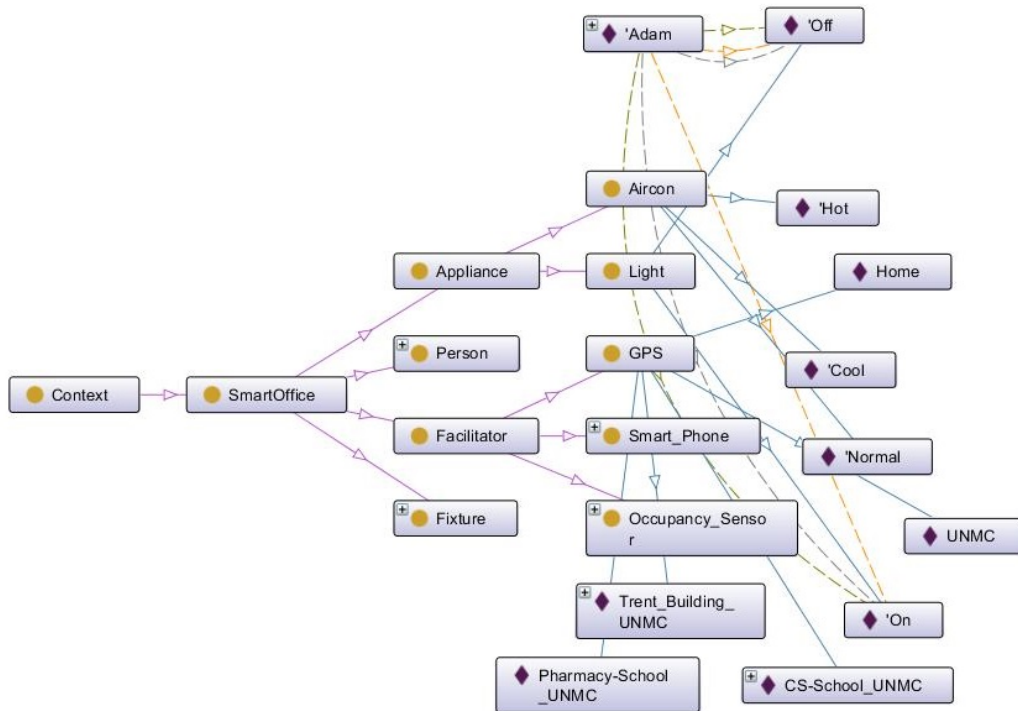


Figure 5.4: Smart office ontology

by the patient care rules. In our Case Study 1, we presented basic preferences, i.e. context-based, which are added before the rule engine starts. However, in case study 2, we will slowly build up the rules as we progress and as new contexts are derived. The ontologies are provided in Figures 5.4 and 5.5. We can observe that both different ontologies have their own rules, if we process them all together, it can drain the battery fast and also can make the rule-engine work slower beside other drawbacks. We will now annotate the rules to provide preferences based on user choices. Doing so will decrease the rules also. Since a designer is well aware of the concepts that are coming from the

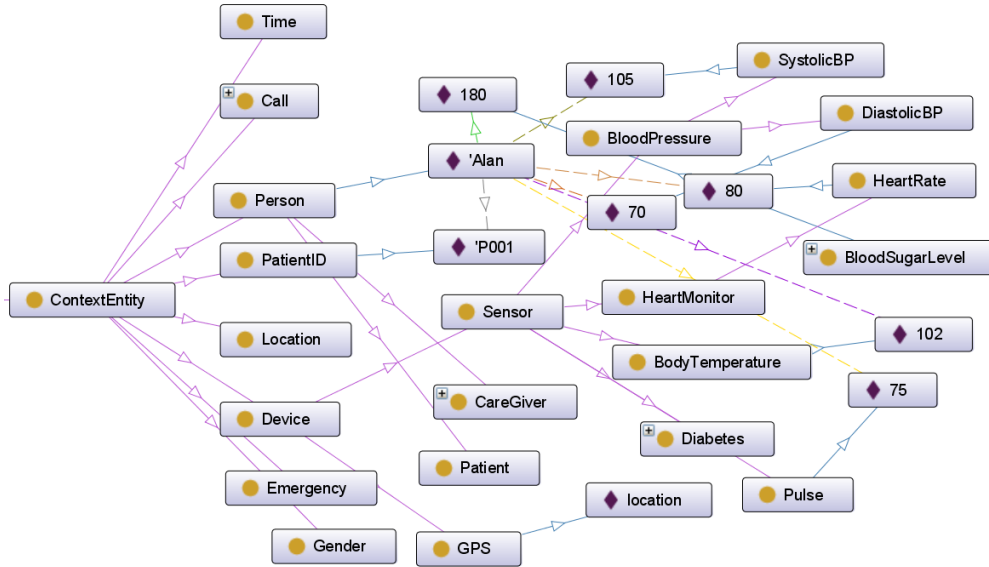


Figure 5.5: Patient care ontology

ontology and on the other hand, he also knows the preferences provided by the user. In our scenario, we assume that a user wants to have a different set of preferences at home and university, i.e. UNMC. He wants to enable the air conditioner whenever he is available in his office. Further he wants to keep a check on his blood pressure and in case of an emergency, he wants to be notified. The preferences for the user at home is to check for blood pressure only and not to include the rules that deal with the UNMC. The rules in Table 5.1 are a small sample of rules. Next, we assume that the user is at home and will move to the office, and we observe the context changes, and so are the rules.

Id	Rule	Identifier
1	Patient(?p), hasBloodPressure(?p, Low) \rightarrow hasSituation(?p, Emergency)	GPS(Home)
2	Patient(?p), hasBloodPressure(?p, Stage2) \rightarrow hasSituation(?p, Emergency)	GPS(Home)
3	Patient(?p), Tell(2,1,hasBloodPressure(?p, Stage2) \rightarrow hasBloodPressure(?p,Stage2)	GPS(Home)
4	Patient(?p), Tell(2,1,hasBloodPressure(?p, Low) \rightarrow hasBloodPressure(?p,Low)	GPS(Home)
5	GPS(?loc), Person(?p) \rightarrow isLocated(?p, ?loc)	-
6	Occupancy_Sensor(?p, ?no) \rightarrow Tell(6,9,hasOccupancy(?p,?yes)	GPS(UNMC)
7	Tell(6,9,hasOccupancy(?p,?yes)) \rightarrow hasOccupancy(?p,?yes)	GPS(UNMC)
8	Occupancy_Sensor(?yes), Person(?p) \rightarrow Tell(6,8,hasOccupancy(?p,?yes))	GPS(UNMC)
9	hasOccupancy(?p,?yes) \rightarrow SwitchAirconFor(?p,?on)	GPS(UNMC)
10	hasOccupancy(?p,?yes) \rightarrow Tell(6,8,hasOccupancy(?p,?yes))	GPS(UNMC)

Table 5.1: Sample rules from two ontologies with identifiers

Facts, CoI and Rules transition

In table 5.2 we elaborate the transition of facts, Context of Interest and how it effect the rules. We start from the user location home towards the UNMC. The CoI column defines the preferences provided by the user, which we assume to be constant throughout the execution and can be changed on demand. The facts column are the high-level contexts received from the sensors. The rules column is populated with the rules as a result of preferences and CoI. Whenever the CoI is found in the facts, it will generate a set of rules as described earlier. We assume a whole cycle from a user Home to UNMC and back to Home.

I	CoI	Facts	CoI found in facts	Rules
1	GPS(Home), GPS(UNMC)	GPS(Home)	Yes	1,2,3,4,5
2	GPS(Home), GPS(UNMC)	\sim GPS(Home), \sim GPS(UNMC)	No	5
3	GPS(Home), GPS(UNMC)	\sim GPS(Home), \sim GPS(UNMC)	No	5
4	GPS(Home), GPS(UNMC)	GPS(UNMC)	Yes	5,6,7,8,9,10
5	GPS(Home), GPS(UNMC)	\sim GPS(Home), \sim GPS(UNMC)	No	5
6	GPS(Home), GPS(UNMC)	GPS(Home)	Yes	1,2,3,4,5

Table 5.2: Preference set transition

The six iterations in Table5.3 shows a whole cycle with a different set of rules selected from a total of ten rules as a sample.

Discussion

In this Paper, we discussed and presented a preference model for the personalisation of resource-bounded context-aware applications based on facts. We also discussed the updated progress of the preferences that are based on the

<p>Iteration 1</p> <p>CoI provided by the user is GPS(Home) and GPS(UNMC) GPS sensor sent facts are GPS(Home) CoI which is GPS(Home) is available in Facts which is GPS(Home) The generated set of rules as a preference set is composed of Patient(?p), hasBloodPressure(?p, Low) \rightarrow hasSituation(?p, Emergency) Patient(?p), hasBloodPressure(?p, Stage2) \rightarrow hasSituation(?p, Emergency) Patient(?p), Tell(2,1,hasBloodPressure(?p, Stage2)) \rightarrow hasBloodPressure(?p, Stage2) Patient(?p), Tell(2,1,hasBloodPressure(?p, Low)) \rightarrow hasBloodPressure(?p, Low) GPS(?loc), Person(?p) \rightarrow isLocated(?p, ?loc)</p>
<p>Iteration 2 and 3</p> <p>CoI provided by the user is GPS(Home) and GPS(UNMC) GPS sensor sent facts are neither GPS(Home) nor GPS(UNMC) CoI is not available in the fact The generated set of rules as a preference set is composed of general rule only GPS(?loc), Person(?p) \rightarrow isLocated(?p, ?loc)</p>
<p>Iteration 4</p> <p>CoI provided by the user is GPS(Home) and GPS(UNMC) GPS sensor sent facts is GPS(UNMC) CoI GPS(UNMC) is available in fact which is GPS(UNMC) The generated set of rules as a preference set is composed of following rules GPS(?loc), Person(?p) \rightarrow isLocated(?p, ?loc) Occupancy_Sensor(?p, ?no) \rightarrow Tell(6,9,hasOccupancy(?p,?yes)) Tell(6,9,hasOccupancy(?p,?yes)) \rightarrow hasOccupancy(?p,?yes) Occupancy_Sensor(?yes), Person(?p) \rightarrow Tell(6,8,hasOccupancy(?p,?yes)) hasOccupancy(?p,?yes) \rightarrow SwitchAirconFor(?p,?on) hasOccupancy(?p,?yes) \rightarrow Tell(6,8,hasOccupancy(?p,?yes))</p>
<p>Iteration 5</p> <p>CoI provided by the user is GPS(Home) and GPS(UNMC) GPS sensor sent facts are neither GPS(Home) nor GPS(UNMC) CoI is not available in the fact The generated set of rules as a preference set is composed of general rule only GPS(?loc), Person(?p) \rightarrow isLocated(?p, ?loc)</p>
<p>Iteration 6</p> <p>CoI provided by the user is GPS(Home) and GPS(UNMC) GPS sensor sent facts are GPS(Home) CoI which is GPS(Home) is available in Facts which is GPS(Home) The generated set of rules as a preference set is composed of Patient(?p), hasBloodPressure(?p, Low) \rightarrow hasSituation(?p, Emergency) Patient(?p), hasBloodPressure(?p, Stage2) \rightarrow hasSituation(?p, Emergency) Patient(?p), Tell(2,1,hasBloodPressure(?p, Stage2)) \rightarrow hasBloodPressure(?p, Stage2) Patient(?p), Tell(2,1,hasBloodPressure(?p, Low)) \rightarrow hasBloodPressure(?p, Low) GPS(?loc), Person(?p) \rightarrow isLocated(?p, ?loc)</p>

Table 5.3: Preference set transition with rules

preferences, along with the rules obtained from different ontologies. Next, we would like to narrow down the preferences further so that it can be applied to the derived context in the working memory. In that case, it will be not only applied before hand, but any context that a user expects to be derived can also opt for preferences.

5.4.2 Case Study 2

Published in [121]. For simplicity sake, we explain in Table 5.4, a different set of rules belonging to different preference indicator, i.e. CS column to focus only on the preferences. The Table explains first some rules which determine low towards high blood pressure stages in four different rules. Similarly, the heart rate is defined in seven different rules to detect whether heart beat is poor or otherwise. Here we assume to add all the blood pressure and heart rate related rules. However, we put preferences to derive different situations. Since the preference set creation mechanism mainly depends on the type of preference applied, and the method of creating a preference set is identical in all case which is selecting all rules from the rule-base having a particular CS indicator (It can be context-based, derived or live preference). Therefore we use context based and derived context to add some rules into the preference set. As it can be observed in the CS of Table 5.4, we have CS indicator as `hasHRCategory(Alan, Poor)`. This implies that whenever some rules infer that Alan has poor heart rate or more formally `hasHRCategory(Alan, Poor)` is added to working memory, then add all rules into preference set corre-

sponding to preference set indicator as applied, i.e. `hasHRCategory(Alan, Poor)`, which will utilise the newly derived context to further infer more context or deduce some output. From the Table, the emergency rules are added only after the `hasHRCategory(Alan, Poor)` is derived from previous rules. Otherwise, these rules will not be processed. Similarly, simple context-based preference, which is applied directly, can be utilised to add rules to preference set directly. In this case, if patient blood pressure drops often, the patient can opt to add the rule directly without the need to first infer, if the blood pressure is low, e.g. `hasBPCategory(Alan,LowBp)`. Using this method, upon startup of the system, the context based preference will be applied first. While rules corresponding to the conditions of poor heart rate will be added only when the previous states infer that the patient heart rate is poor. The `hasLocation(Alan, UNMC)` is live preference and will be invoked only when the GPS sensor senses the location of Alan at UNMC.

Category	m	Corresponding rule	F	CS
Low BP	1	<code>Person(?p), hasSystolicBloodPressure(?p,?sbp), hasDiastolicBloodPressure(?p, ?dbp), lessThan(?sbp, '90), lessThan(?dbp,60) → hasBPCategory(?p,LowBP)</code>	D	-
Normal	1	<code>Person(?p),hasSystolicBloodPressure(?p,?sbp), hasDiastolicBloodPressure(?p, ?dbp), greaterThan(?sbp,90), greaterthan(?dbp,60), lessThan(?sbp,120), less-</code>	D	-

		Than(?dbp,80) \rightarrow hasBPcategory(?p,Normal)		
Pre high	1	Person(?p), hasSystolicBloodPressure(?p,?sbp), hasDiastolicBloodPressure(?p, ?dbp),greaterThan(?sbp,120), greaterThan(?dbp,80),lessThan(?sbp,140), less- Than(?dbp,90) \rightarrow hasBPcategory(?p,PreHigh)	D	-
High	1	Person(?p), hasSystolicBloodPressure(?p,?sbp), hasDiastolicBloodPressure(?p, ?dbp), greaterThan(?sbp,140), greaterThan(?dbp,90) \rightarrow hasBPcategory(?p, HighBP)	D	-
Category	m	Corresponding rule	F	CS
Athlete	1	Person(?p), hasHeartRate(?p,?hrt), greaterThan(?hrt,48), lessThan(?hrt,55) \rightarrow hasHRcategory(?p, Athlete)	D	-
Excellent	1	Person(?p), hasHeartRate(?p,?hrt), greaterThan(?hrt,54), lessThan(?hrt,62) \rightarrow hasHRcategory(?p,Excellent)	D	-
Good	1	Person(?p), hasHeartRate(?p,?hrt), greaterThan(?hrt,61), lessThan(?hrt,66) \rightarrow hasHRcategory(?p,Good)	D	-
Above Average	1	Person(?p), hasHeartRate(?p,?hrt), greaterThan(?hrt,65), lessThan(?hrt,71) \rightarrow hasHRcategory(?p,AboveAverage)	D	-

Average	1	Person(?p), hasHeartRate(?p,?hrt), greaterThan(?hrt,70), lessThan(?hrt,75) → hasHRCategory(?p,Average)	D	-
Below Average	1	Person(?p), hasHeartRate(?p,?hrt), greaterThan(?hrt,74),lessThan(?hrt,82) → hasHRCategory(?p,BelowAverage)	D	-
Poor	1	Person(?p), hasHeartRate(?p,?hrt), greaterThan(?hrt,81) → hasHRCategory(?p,Poor)	D	-
Category	m	Corresponding rule	F	CS
Emergency	2	Patient(?p), hasBPCategory(?p,HighBP), hasHRCategory(?p,Poor)→ hasSituation (?p,Emergency)	D	hasHRCategory (Alan, Poor)
Emergency	2	Patient(?p), hasBPCategory(?p,PreHigh), hasHRCategory(?p,Poor)→ hasSituation (?p,Emergency)	D	hasHRCategory (Alan, Poor)
Emergency	2	Patient(?p),hasBPCategory(?p,Normal), hasHRCategory(?p,Poor)→ hasSituation (?p,Emergency)	D	hasLocation (Alan,Unmc)
Emergency	2	Patient(?p),hasBPCategory(?p,LowBp), hasHRCategory(?p,Poor) → hasSituation (?p,Emergency)	D	hasBPCategory (Alan,LowBp)

Non Emergency	1	Patient(?p),hasBPCategory(?p,Normal), hasHRCategory(?p,Average) \rightarrow \sim hasSituation (?p,Emergency)	D	hasLocation (Alan,UNMC)
Non Emergency	1	Patient(?p),hasBPCategory(?p,Normal), hasHRCategory(?p,AboveAverage) \rightarrow \sim hasSituation (?p,Emergency)	D	hasLocation (Alan,UNMC)
Non Emergency	1	Patient(?p),hasBPCategory(?p,Normal), hasHRCategory(?p,Good) \rightarrow \sim hasSituation (?p,Emergency)	D	hasLocation (Alan,UNMC)

Table 5.4: Blood pressure, heart rate rules and example rules

While executing the sample rules initially, the preference set will have only single rule, i.e. with CS of `hasBPCategory(Alan, LowBP)`. After further execution and when the Heart Rate Monitor derive a new context of `hasHRCategory(Alan, Poor)`. The preference set will have a few more rules which belong to derived preference with CS `hasHRCategory(Alan, Poor)`. Similarly, rest of the rules with CS `hasLocation(Alan, UNMC)`, will only be added when the user is physically at UNMC, and the device(GPS) detects the user location at UNMC. In that way the preference set is slowly building up instead of putting all the rules at once it only adds rules which the user has selected based on the CS and preference mechanism.

5.4.3 Case Study 3

Published in [122]. We consider a system consisting of several agents, including a person agent (Agent 1 represented by a smartphone) who is a user and may change his location detected by the GPS embedded into his smartphone. The user is also known to have his Blood pressure issues, which is monitored by the BP device (Agent 2) and has a heart rate monitor enabled (Agent 3). The user casually visits the hospital for the check-up, and person agent can interact with Out Patient handling agent (Agent 4, located at Hospital). The user also has some preferences for his office, which is located at UNMC. The office has an occupancy sensor (Agent 5), which can detect if the user is in the office or not.

Context-based Preferences

As mentioned above, the user is not static, and he changes his location. As he arrives at the hospital, his location is detected and processed to derive a new context being a patient. We will use this derived context to make a preference set for him at the hospital, which will illustrate how the sensed/externally received context-based preference as well as derived-context based preference work together to minimise the load on the agent's inference engine by minimising the number of rules while still receiving the desired results. The rules in Table 5.5 are some example rules that are used to design Agent 1. The initial facts provided to the system are *PatiendID(101)* and *hasPatientID(Alan,101)*, and the GPS sensor detects the location and that location

is added to the agent's working memory as a fact. Once the COI is defined, the system checks and separate the COI from $_COI$. The $_COI$ is put aside for the later use once the system starts working. As a result, the Table 5.6 shows us details on the rules that are supposed to be in the preference set for a given set of user-provided preferences. In Table 5.6, we elaborate on the transition of facts, Context of Interest, and how it affects the rules. We assume the initial location of the user is at his Home. Later on, the user visits the smart hospital and accordingly, his location is detected, which in turns deduce that the user is a patient. Accordingly, the derived-context used as a preferred context that helps to generate a new set of rules by replacing the existing rules to be used in the agent's inference engine.

Rule-based Preferences

It is always possible that a conflict occurs between the rules, and to resolve it, we assign priorities (column m in Table 5.5) to the rules. The rule priorities give one rule preference over another rule. In this case study, we deliberately made a scenario where according to the facts we can have two different rules generating contradictory outcome as $hasSituation(Alan, Emergency)$ and $\sim hasSituation(Alan, Emergency)$. Which if not handled, can derive an unwanted conclusion. Therefore, we assigned the priorities to rules, as a part of defeasible reasoning, and in the scenario described below the rules, R1 and R2 are assigned priority 3, while R5 has priority 1. Since R1 and R2 having higher priority than that of R5, the preference will be given to R1 and R2

Id	m	Rule	Identifier
R1	3	Patient(?p), hasBloodPressure(?p, Low) \longrightarrow hasSituation(?p, Emergency)	_Patient(Alan)
R2	3	Patient(?p), hasBloodPressure(?p, High) \longrightarrow hasSituation(?p, Emergency)	_Patient(Alan)
R3	2	Tell(2,1,hasBloodPressure(?p, High) \longrightarrow hasBloodPressure(?p,High)	_Patient(Alan)
R4	2	Tell(2,1,hasBloodPressure(?p, Low) \longrightarrow hasBloodPressure(?p,Low)	_Patient(Alan)
R5	1	Patient(?p), hasHeartRate(?p, Normal) \longrightarrow \sim hasSituation(?p, Emergency)	_Patient(Alan)
R6	2	Tell(3,1,hasHeartRate(?p, Normal) \longrightarrow hasHeartRate(?p,Normal)	_Patient(Alan)
R7	1	Person(?p), GPS(?loc) \longrightarrow hasLocation(?p, ?loc)	-
R8	2	hasLocation(?p, Hospital), PatientID(101), hasPID(?p,101) \longrightarrow Patient(?p)	-
R9	2	Patient(?p), hasReason(?p, ?r), MedicalReason(?r) \longrightarrow isOutPatient(?p,?r)	_Patient(Alan)
R10	2	isOutPatient(?p, ?r) \longrightarrow Tell(1, 4, isOutPatient(?p, ?r))	_Patient(Alan)
R11	2	Tell(5,1,hasOccupancy(?p, Yes)) \longrightarrow hasOccupancy(?p,Yes)	GPS(UNMC)
R12	2	hasOccupancy(?p,Yes)) \longrightarrow Tell(1, 6, hasAir-con(?p,On))	GPS(UNMC)

Table 5.5: Some example rules of Agent 1

over R5. Thus, avoiding any unwanted outcome. A more detailed discussion on defeasible reasoning can be found in [107].

5.4.4 Case Study 4

In this case study, a detailed study is provided. The case study focuses on a one-day scenario of a user. The scenario follows a user-assisted living style for a whole day. The case study includes scenarios such as home, office, hospital, market and smart fridge. Also, it shows how preferences can reduce

System status	COI	_COI	Facts in WM
Initial information	GPS(UNMC)	Patient(Alan)	PatientID(101), hasPatientID(Alan, 101)
Iterations of the system case scenario, where a user moves to different locations at different times with preferences enabled are GPS(UNMC) and Patient(Alan)			
User location	Derived facts	Preference indicator found in WM	Corresponding subset of rules
GPS(Home)	–	No	R7, R8
GPS(UNMC)	–	GPS(UNMC)	R7, R8, R11, R12
GPS(Hospital)	hasLocation(Alan, Hospital) Patient(Alan)	Patient(Alan)	R1,R2,R3,R4, R5,R6,R7,R8,R9,R10

Table 5.6: Preference set transition

rule-base size. Before moving to the case study, a detailed explanation of the system setups, different environment, ontologies, agents used and their respective rules are provided for reference.

System Setup

The system is based on different agents, the majority of the agents are Android-based agents with their inference engine and memory. Some agents are only capable to sense data and send it, and does not have any inference ability. In order to cover full scenario details, some agents are simulated such as lighting sensor. Overall there are five ontologies, having 29 different agents. There are total of 167 rules, working together to accomplish a variety of tasks without the user intervention. Furthermore, rules corresponding to their agents, along with other details, are provided in the appendix section of this thesis.

Smart home	Smart office	Smart HealthCare	
Authorization Sensor Motion Detector Light Sensor Aircon Controller Home Controller Sensor Temperature Sensor Door Control Sensor Gas Leak Detector Smoke Sensor	Smart Task Manager Employee Smart Chair Light Lamp Windows Blinds Office Temperature Office Aircon	Patient Care Agent Blood Pressure Monitor Diabetes Meter Fever Monitor Pulse Monitor Emergency Monitor GPS sensor	
	<th>Smart Market</th> <th>Smart Fridge</th>	Smart Market	Smart Fridge
	Smart Shopping cart	Egg Sensor Milk Sensor Butter Sensor Water Dispenser Fridge Door Monitor	

Figure 5.6: Ontologies with corresponding agents

Ontologies and Agents

Figure 5.6 provides all the ontologies and their corresponding agents. The grey heading defines the name of the ontologies, while the list below gives the agents used in the ontology.

The patient care ontology onto-graphs is presented for reference at Figure 5.7.

As discussed in chapter 4, the Onto-HCR tool is used to access the ontologies and translate them into the corresponding horn-clause rules. The five different ontologies are translated into a single file and amended for preferences on the rules and priorities. The resultant rule-base comprises of more than 150 rules.

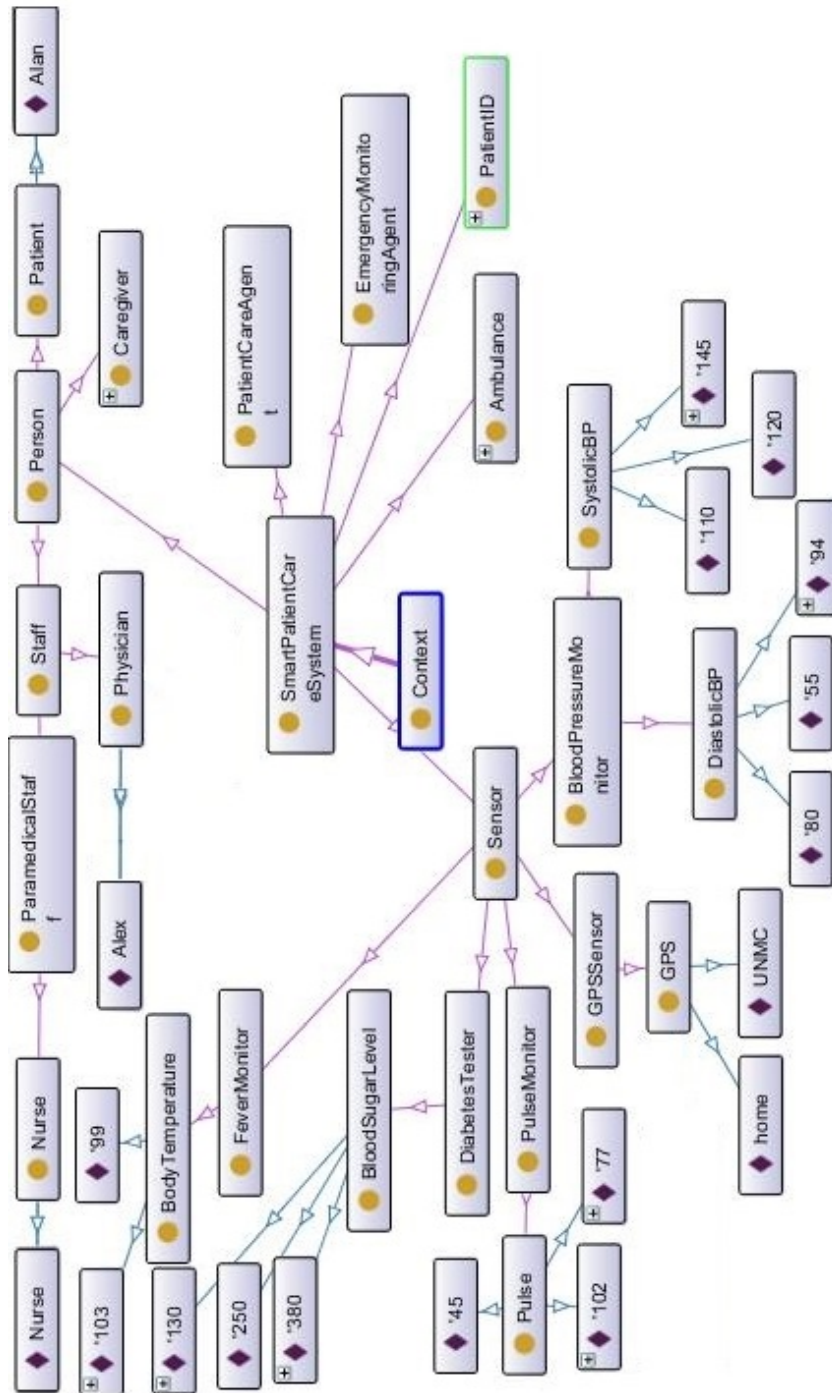


Figure 5.7: Patient Care Ontology

Scenarios

The case study focuses on the normal routine of a user. System response is checked on a different location, e.g. home, office, market and hospital. In the next section, each of the following locations is briefly discussed and also the role of a specific agent is briefed. In this case study, preferences are provided on the rules which dynamically change the system behaviour.

Agents Details

- **Smart Home:** Smart home provides services that are specific to home use. It has nine different agents, which works together to serve the user in a better way.
 1. *Authorization sensor:* It checks if the user is authorized to use the services or not.
 2. *Motion Detector:* This agent detects the motion in different rooms, based on the detection it can operate further.
 3. *Light Sensor:* It works with lights, especially turning the lights on and off.
 4. *Aircon Controller:* This sensor controls the working of the air conditioner, to keep the temperature at a comfortable level.
 5. *Home Controller Sensor:* This sensor checks the occupancy and authorization of the user at home.

6. *Temperature Sensor*: Temperature sensor sense the temperature in the room and accordingly send it to the air conditioner for adjusting to the comfortable level.
 7. *Door Control Sensor*: This sensor is attached to the doors, and it can open/close the door, OR it can show if the door is opened.
 8. *Gas Leak Detector*: It detects the gas leakage, and if the leakage is found, it can alert the user.
 9. *Smoke Sensor* : Just like a Gas detector, it detects the smoke and fire the alarm in case of smoke is detected
- **Smart Office**: Smart office facilitates the user in the office by keeping the environment comfortable according to its rules.
 1. *Smart Task Manager*: It provides the user the tasks list or more like things to do memo at the office.
 2. *Employee*: It checks for the user employee status at the office.
 3. *Smart Chair*: It detects if the user is sitting in the office, it also reminds the user to change posture or walk in case it detects that the user is sitting for a long.
 4. *Light Lamp*: It controls the lights at the office.
 5. *windows blinds*: It controls the windows blinds to open or close them.

6. *Office temperature*: it checks the office temperature, and in case the temperature is not comfortable, the air con agents adjust it.
 7. *Office Aircon* : it maintains the office temperature.
- **Smart Health Care**: This is one of the main system, responsible for checking the user's health and in case of emergency, it can alert the emergency services.
 1. *Patient care agent*: This agent is the main agent to facilitate the user health condition. It checks for different condition and accordingly, for example, in case of emergency, it can alert the emergency service.
 2. *Blood Pressure Monitor*: It checks for the user blood pressure and sends it to the Patient care agent.
 3. *Diabetes meter*: It checks for the diabetes level and sends it to the Patient care agent.
 4. *Fever Monitor*: It checks for the Fever level and sends it to the Patient care agent.
 5. *Pulse Monitor*: It checks for the user pulse and sends it to the Patient care agent.
 6. *Emergency Monitor* : This agent is responsible for handling the emergency situation when the user is in a critical situation.

7. *GPS sensor*: GPS sensor finds the user location and when required, send it to other agents. For example, in an emergency situation, it can send the GPS to the ambulance.
- **Smart Market**: Its main task is to remind the user of the required grocery at home; it is attached to the Smart Fridge.
 1. *Smart Shopping Cart*: It mainly receives instructions from the Smart fridge, and let the user reminds about the grocery required.
 - **Smart Fridge**: It monitors different items in the fridge and their quantity.
 1. *Egg sensor* : It checks for the egg quantities in the fridge.
 2. *Milk Sensor*: It checks the milk if the user is short on milk; it notifies the user when the user is in the market.
 3. *Butter Sensor* : It checks for the butter quantity.
 4. *Water Dispenser*: It checks if the water dispenser has enough water or not.
 5. *Fridge Door Monitor* : It checks for the door open/close status.

Scenario Execution Smart Patient, Home And Office

In this section, we execute different case scenarios to understand the system behaviour, with preferences applied to the rule-base. The preference in these cases is applied on different levels. The scenario executed provides a user

who has medical issues, and the user goes to the office, and have preferences set-up for his/her medical, home and office scenario. The user has opted out from the market and smart fridge execution.

Dry Run Analysis

Since the framework works in a collaborative way to accomplish a goal. In the Tables 5.8, 5.9, 5.10, an execution of scenario for smart patient is provided stepwise. It takes a total of 23 steps and seven different agents to respond to an emergency situation. The preferences are set to turn on whenever an emergency is deduced. For normal condition or not emergency cases, it does not inform the doctor or caregiver. In the dry run, the findings are quite impressive. In the Table 5.7, agent-wise distribution is elaborated. It can be observed that the first column refers to the Agent name and its corresponding number. Associated rules refer to rules associated with a particular agent. No of rules shows the total number of rules a particular agent has. The fourth column represents the number of rules after certain preferences are provided. The last column shows the number of processable rules in percentage form. The last row shows consolidated results for all agents.

Agent Name and Number	Associated Rules	No. of Rules	Rules After Preference	Rules in %
Patient Care Agent-1	1-42	42	29	69
Blood Pressure Agent-2	43-51	9	2	22
Diabetes Meter Agent-3	52-61	10	2	20
Fever Monitor Agent-4	62-69	8	2	25
Pulse Monitor Agent-5	70-78	9	2	22
Emergency Monitor Agent-6	79-83	5	5	100
GPS Sensor Agent-7	84-85	2	2	100
Authorization Sensor Agent-8	86-88	3	3	100
Motion Detector Agent-9	89-95	7	7	100
Light Sensor Agent-10	96-99	4	4	100
Aircon Controller Agent-11	100-105	6	6	100
Home Controller Sensor Agent-12	106-109	4	0	0
Temperature Sensor Agent-13	110-114	5	5	100
Door Control Sensor Agent-14	115	1	0	0
Gas Leak Detector Agent-15	116	1	0	0
Smoke Sensor Agent-16	117	1	0	0
Smart Task Manager Agent-17	118-120	3	0	0
Employee Agent-18	121-127	7	1	14

Agent Name and Number	Associated Rules	No. of Rules	Rules After Preference	Rules in %
Smart Chair Agent-19	128-130	3	0	0
Light Lamp Agent-20	131-136	6	0	0
Window Blinds Agent-21	137-143	7	0	0
Office Temperature Agent-22	144-149	6	6	100
Office Aircon Agent-23	150-153	4	0	0
Smart Shopping Cart Agent-24	154-159	6	0	0
Egg Container Sensor Agent-25	160-161	2	0	0
Milk Pack Quantity Sensor Agent-26	162-163	2	0	0
Butter Cube Quantity Sensor Agent-27	164-165	2	0	0
Fridge Door Sensor Agent-28	166-167	2	0	0
	Total	167	76	45%

Table 5.7: Complete Agent Wise Rules and Reductions in Rules

In the Table 5.7, it is observed that the number of rules required to handle the emergency situation case can be completed with 44 different rules from 7 different agents, which represent that the remaining rules are either redundant or give results which are not required, hence removed successfully

with the use of preferences.

In the smart home scenario, a location-based preference is enabled on rules. The GPS senses the location and based on the sensed context; a preference set is defined. The complete rules for the ontology is defined in the Appendix A. Stepwise execution is provided in the Tables 5.11, 5.12, 5.13.

In the smart office scenario combination of preferences work together to get the employee to adjust only air-conditioner as per the preference defined. The execution is defined in Table 5.14, while a complete set of rules are provided in Appendix A.

5.4.5 Case Study 5

In this case study, we tested the system with the worst possible rules. In this experiment, we have used five different rules that are enough to deduce some goal. Then the same rules are repeated numerous times so that we have duplicate rules, rules with the same priority, same variable instantiations, and matching rule for various times to name a few of the possibilities. The purpose of testing the system with such rules is to check if the system can accurately produce the same results without preference and then when the preferences are enabled, how many rules does it take to reach the goal. The goal is already known; this test is conducted to check for the working mechanism in both cases. The output is provided in Figure 5.8 with preferences enabled and Figure 5.9 with preference disabled. Figure 5.9 shows

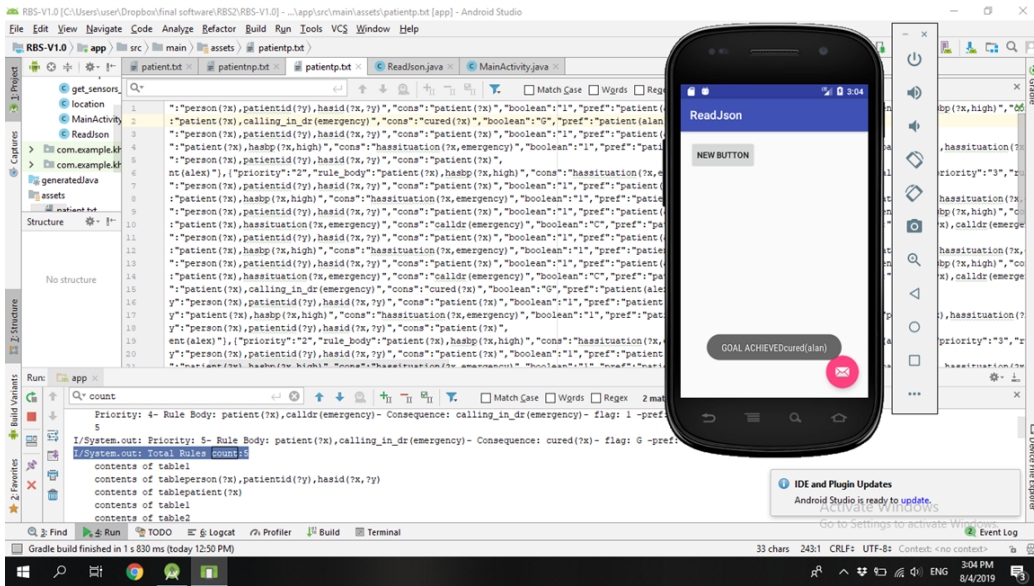


Figure 5.8: Preferences enabled

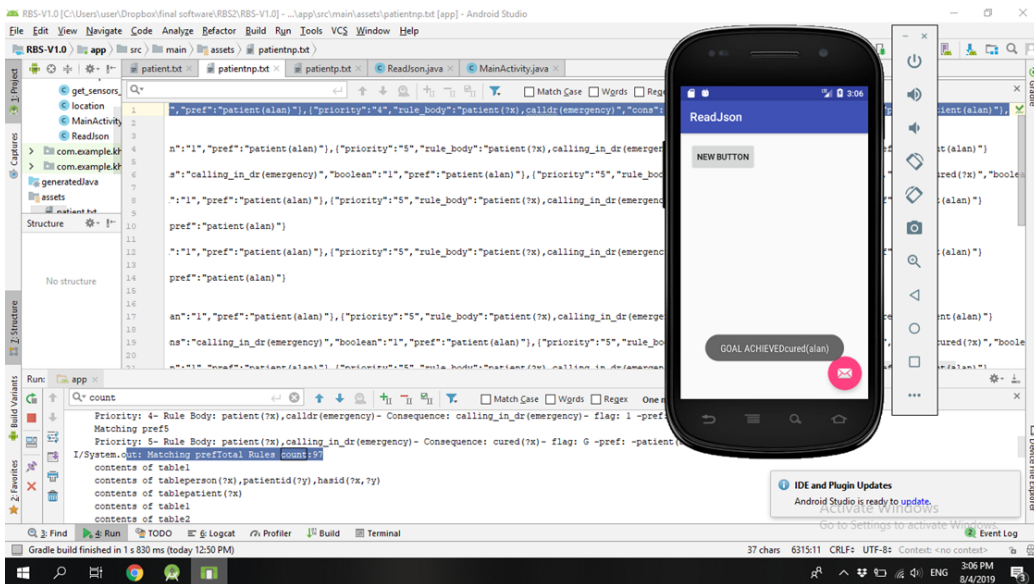


Figure 5.9: Preferences disabled

that the rule-engine has to go through all the rules in order to generate results which can be derived with only five rules. It has to rematch, evaluate all the rules, variable bindings and produce the goal of `cured(alan)`. Figure 5.8 shows that when we enabled the preferences, and the carefully applied context-based preferences are defined, then the rule-engine before go through rule-base make a subset of the eligible rules and picks only five qualified rules based on the preferences for processing. It can be seen in the Figures that both produce the same results while without preferences the number of rules is ninety-seven, while with preferences the number of rules is reduced to only five.

5.4.6 Case Studies Results

In case study 1, we have a total of 10 rules. Average used rules are 3.1. The average reduction is 69%. Maximum reduction was found to be 90%, while the minimum reduction was 60%. In case study 2, we have a total of 18 rules. Average rules used were 3. The average reduction in rules was found to be 28%. Maximum reduction was 44% while the minimum reduction was 22%. In case study 3, we have a total of 12 rules with average rules reduction at 55%, and the maximum reduction was found to be 83.3%, and the minimum reduction was found to be 16.6%. Case study 4, which is the most detailed case study in all these studies, shows a total reduction in rules to be 55%, where the total number of rules is 167. Case study 5, is slightly different from the rest of the case studies, instead of reduction in rules we tested it with five

qualified rules of the same nature to check the worst case of the rule-engine. The rule-engine provided the same results with and without preferences. Only the number of processed rules changed with and without preferences as described in the case study. Overall the results are quite satisfactory in different scenarios provided.

5.5 Discussion

In this chapter, we have extended our previous framework to include preferences. Preferences are used to achieve two main purposes. First, it can reduce the size of rule-base significantly. Secondly, it uses the same feature to provide the user with personalized services, for example, having office related rules in rule-base when the user is in office. In the preferences, we have covered all the instances when the context can play a role to change the shape of the rule-base. It can be applied before the rule-engine starts and also in run time while a new context is derived or a live context change is detected. The chapter covers different case studies with different preferences and applied preferences. Also, we covered logical case studies, dry run analysis and implementation of rules. The results with and without preferences show the same results but with a different number of processed rules. The results further attest the efficiency of the system. The execution tracing of a single scenario takes much effort. Therefore the dry run is designed in a way to provide the concrete steps and omitting any unnecessary steps involved. The execution

halts after it reaches its final state. The output received in case study 4, is satisfactory, and the same results are obtained without preferences, which shows the correctness of the system with and without preferences is the same. The preference here plays an important role and reduce the redundancy that may arise when unwanted rules are processed every single time; the system is working. Since this thesis covered major technical areas to deploy a framework, there is always room for improvement. In the next chapter, the thesis is concluded along with suggested future direction for research.

step	Patient Care Agent(1)	Action	Blood Pressure Monitor(2)	Action
1	PatientID(01), Person(Alan), hasPatientID(Alan, 01) — Patient(Alan)	R1	DiastolicBP(90), Person(Alan), SystolicBP(130), hasDiastolicBP(Alan, 90), hasSystolicBP(Alan, 130), greaterThanOrEqual(90, 85), greaterThanOrEqual(130, 125) — hasBloodPressure(Alan, High)	R45
2	idle	-	DiastolicBP(90), Person(Alan), SystolicBP(130), hasDiastolicBP(Alan, 90), hasSystolicBP(Alan, 130), greaterThanOrEqual(90, 85), greaterThanOrEqual(130, 125) — Tell(2,1, hasBloodPressure(Alan, High))	R51
3	PatientID(01), Person(Alan), hasPatientID(Alan, 01) — Patient(Alan), Tell(2, 1, hasBloodPressure(Alan, High))	copy		
4	PatientID(01), Person(Alan), hasPatientID(Alan, 01) — Patient(Alan), hasBloodPressure(Alan, High)	R36		
5	PatientID(01), Person(Alan), hasPatientID(Alan, 01) — Patient(Alan), hasBloodPressure(Alan, High), Tell(3, 1, hasDBCateory(Alan, High))	copy		
6	PatientID(01), Person(Alan), hasPatientID(Alan, 01) — Patient(Alan), hasDBCateory(Alan, High)	R29		
7	PatientID(01), Person(Alan), hasPatientID(Alan, 01) — Patient(Alan), hasBloodPressure(Alan, High), hasDBCateory(Alan, High), Tell(4, 1, hasFever'(Alan, High))	copy		
8	PatientID(01), Person(Alan), hasPatientID(Alan, 01) — Patient(Alan), hasBloodPressure(Alan, High), hasDBCateory(Alan, High), hasFever'(Alan, High)	R30		
9	PatientID(01), Person(Alan), hasPatientID(Alan, 01) — Patient(Alan), hasBloodPressure(Alan, High), hasDBCateory(Alan, High), hasFever'(Alan, High), Tell(5, 1, hasPulseRate'(Alan, High))	copy		
10	PatientID(01), Person(Alan), hasPatientID(Alan, 01) — Patient(Alan), hasBloodPressure(Alan, High), hasDBCateory(Alan, High), hasFever'(Alan, High), hasPulseRate'(Alan, High)	R33		
11	PatientID(01), Person(Alan), hasPatientID(Alan, 01) — Patient(Alan), hasBloodPressure(Alan, High), hasDBCateory(Alan, High), hasFever'(Alan, High), hasPulseRate'(Alan, High), hasSituation(Alan, Emergency)	R5		
12	PatientID(01), Person(Alan), hasPatientID(Alan, 01) — Patient(Alan), hasBloodPressure(Alan, High), hasDBCateory(Alan, High), hasFever'(Alan, High), hasPulseRate'(Alan, High), hasSituation(Alan, Emergency), Tell(1,6, hasSituation(Alan, Emergency))	R42		
13	idle	R44		idle
14	idle	-	idle	idle
15	idle	-	idle	idle

Table 5.8: Execution of Smart Patient (A)

Step	Diabetes (Agent 3)	Action	Fever (Agent 4)	Action	Pulse Monitor (Agent 5)	Action
1	BloodSugarLevel(250), Person(Alan), greaterThanOrEqual(250,130), lessThan(250,300) -hasDBCategory(Alan, High)	R56	BodyTemperature(102), Person(Alan), hasBodyTemperature(Alan,102), greaterThanOrEqual(Alan, 99), lessThan(102, 103) -hasFever(Alan, High)	R65	Person(Alan), hasPulse(Alan, 115), greaterThanOrEqual(115,100), lessThan(115,120), hasPulseRate(Alan, High)	R70
2	BloodSugarLevel(250), Person(Alan), greaterThanOrEqual(250,130), lessThan(250, 300), -Tell(3,1,hasDBCategory(Alan, High))	R60	BodyTemperature(102), Person(Alan), hasBodyTemperature(Alan,102), greaterThanOrEqual(102, 99), lessThan(102, 103) -Tell(4,1,hasFever(Alan, High))	R67	Person(Alan), hasPulse(Alan, 115), greaterThanOrEqual(115,100), lessThan(115, 120) Tell(5,1,hasPulseRate(Alan, High))	R77
3	idle	idle	idle	idle	idle	idle
4	idle	idle	idle	idle	idle	idle
5	idle	idle	idle	idle	idle	idle
6	idle	idle	idle	idle	idle	idle
7	idle	idle	idle	idle	idle	idle
8	idle	idle	idle	idle	idle	idle
9	idle	idle	idle	idle	idle	idle
10	idle	idle	idle	idle	idle	idle
11	idle	idle	idle	idle	idle	idle
12	idle	idle	idle	idle	idle	idle
13	idle	idle	idle	idle	idle	idle
14	idle	idle	idle	idle	idle	idle
15	idle	idle	idle	idle	idle	idle
16	idle	idle	idle	idle	idle	idle
17	idle	idle	idle	idle	idle	idle
18	idle	idle	idle	idle	idle	idle
19	idle	idle	idle	idle	idle	idle
20	idle	idle	idle	idle	idle	idle
21	idle	idle	idle	idle	idle	idle
22	idle	idle	idle	idle	idle	idle
23	idle	idle	idle	idle	idle	idle

Steps	Emergency Agent(6)	Action	GPS(7)	Action
15	Patient(Alan), Physician(Alex) —Tell (1, 6, hasSituation(Alan, Emergency))	copied	idle	
16	Patient(Alan), Physician(Alex) —hasSituation(Alan, Emergency)	R79	idle	
17	Patient(Alan), Physician(Alex), hasGPSLocation(Alan, UNMC) —hasSituation(Alan, Emergency), Ask (6, 7, hasGPSLocation(Alan, ?loc))	R81	idle	
18	idle	-	GPSLocation(UNMC), Person(Alan) —Ask 6, 7, hasGPSLocation'(Alan, ?loc)	copy
19	idle	-	GPSLocation(UNMC), Person(Alan) —Ask (6, 7, hasGPSLocation'(Alan, ?loc)), hasGPSLocation(Alan, UNMC)	R84
20	idle	-	GPSLocation(UNMC), Person(Alan) —Ask (6, 7, hasGPSLocation'(Alan, ?loc)), hasGPSLocation(Alan, UNMC), 'Tell (7, 6, hasGPSLocation'(Alan, UNMC))	R85
21	Patient(Alan), Physician(Alex) —hasSituation(Alan, Emergency), Tell (7, 6, hasGPSLocation(Alan, UNMC))	copy	idle	-
22	Patient(Alan), Physician(Alex) —hasSituation(Alan, Emergency), hasGPSLocation(Alan, UNMC)	R81	idle	-
23	Patient(Alan), Physician(Alex) —hasSituation(Alan, Emergency), hasGPSLocation(Alan, UNMC), isDiagnosedBy(Alan, Alex)	R82	idle	-

Table 5.10: Execution of Smart Patient (C)

steps	Authorization Agent(8)	Action
1	AuthorizationID(A01), BiometricSensor(B01), Person(Alan), hasAuthorizationID(Alan, A01), hasBiometricID(Alan, B01)——	r86
2	AuthorizationID(A01), BiometricSensor(B01), Person(Alan), hasAuthorizationID(Alan, A01), hasBiometricID(Alan, B01)—— isAuthorizedPerson(Alan, Yes)	r87
3	AuthorizationID(A01), BiometricSensor(B01), Person(Alan), hasAuthorizationID(Alan, A01), hasBiometricID(Alan, B01)—— isAuthorizedPerson(Alan, Yes)	r88
4	idle	idle
5	idle	idle
6	idle	idle
7	idle	idle
8	idle	idle
9	idle	idle
10	idle	idle
11	idle	idle
12	idle	idle
13	idle	idle

Table 5.11: Execution of Smart Home (A)

steps	Motion Detector(9)	Action	Light Sensor(10)	Action
1	idle	idle	idle	idle
2	Room(Bedroom), MotionDetector(true), isAvailable(Alan, Bedroom) —Tell(8, 9, isAuthorizedPerson(Alan, Yes))	copy	idle	idle
3	Room(Bedroom), MotionDetector(true), isAvailable(Alan, Bedroom) —isAuthorizedPerson(Alan, Yes))	r94	idle	idle
4	Room(Bedroom), MotionDetector(true), isAvailable(Alan, Bedroom) —isAuthorizedPerson(Alan, Yes))	r92	idle	idle
5	Room(Bedroom), MotionDetector(true), isAvailable(Alan, Bedroom) —isAuthorizedPerson(Alan, Yes)), hasOccupancy(Alan, Yes)	r89	idle	idle
6	Room(Bedroom), MotionDetector(true), isAvailable(Alan, Bedroom) —isAuthorizedPerson(Alan, Yes)), hasOccupancy(Alan, Yes)	r90	—Tell(9, 10, hasOccupancy(Alan, Yes)	rule copied to WM
7	Room(Bedroom), MotionDetector(true), isAvailable(Alan, Bedroom) —isAuthorizedPerson(Alan, Yes)), hasOccupancy(Alan, Yes)	r91	— hasOccupancy(Alan, Yes)	r100
8	Room(Bedroom), MotionDetector(true), isAvailable(Alan, Bedroom) —isAuthorizedPerson(Alan, Yes)), hasOccupancy(Alan, Yes)	r93	idle	idle
9	isAvailable(Alan, Bedroom), Room(Bedroom), isAvailable(Alan, Bedroom)	r95	— hasOccupancy(Alan, Yes), Tell(9, 10, AvailableAt(Alan, Bedroom))	copy
10	idle	idle	— hasOccupancy(Alan, Yes), AvailableAt(Alan, Bedroom))	r96
11	idle	idle	— hasOccupancy(Alan, Yes), AvailableAt(Alan, Bedroom), hasLightStatus(Alan, On) ⁷⁰	r99
	Chapter 5			
12	idle	idle	idle	idle
13	idle	idle	idle	idle

Table 5.12: Execution of Smart Home (B)

steps	Air Con Con- troller(11)	action	Temperature Sen- sor(13)	Action
1	idle	idle	idle	idle
2	idle	idle	idle	idle
3	idle	idle	idle	idle
4	idle	idle	idle	idle
5	idle	idle	idle	idle
6	idle	idle	idle	idle
7	idle	idle	idle	idle
8	—Tell(9, 11, hasOccupancy(Alan, Yes)	copy	idle	idle
9	—hasOccupancy (Alan, Yes)	r102	idle	idle
10	— hasOccu- pancy(Alan, Yes), Tell(9, 11, Avail- ableAt(Alan, Bed- room),	copy	Temperature(30)— hasTemp(30, Hot)	r113
11	— hasOccu- pancy(Alan, Yes), AvailableAt(Alan, Bedroom)	r105	Temperature(30)— hasTemp(30, Hot)	r112
12	—hasOccupancy(Alan Yes), isAvail- ableAt(Alan, Bed- room), Tell(13, 11, hasTemp(30, Hot)	copy	idle	idle
13	—hasOccupancy(Alan Yes), Avail- ableAt(Alan, Bed- room), hasTemp(30, Hot) —hasAircon- Status(Bedroom,On)	r100	idle	idle

Table 5.13: Execution of Smart Home (C)

step	Employee (19)	action	OfficeTemperature (23)	action	Office Aircon (24)	action
1	EmployeeID(0001), Person(Alan) — Employee(Alan)	r121	idle	idle	idle	idle
2	EmployeeID(0001), Person(Alan) — Employee(Alan)	r122	idle	idle	idle	idle
3	EmployeeID(0001), Person(Alan) — Employee(Alan)	r123	idle	idle	idle	idle
4	EmployeeID(0001), Person(Alan) — Employee(Alan)	r124	idle	idle	idle	idle
5	EmployeeID(0001), Person(Alan) — Employee(Alan)	r125	idle	idle	idle	idle
6	EmployeeID(0001), Person(Alan) — Employee(Alan)	r126	idle	idle	idle	idle
7	EmployeeID(0001), Person(Alan) — Employee(Alan)	r127	OfficeTemp(28) —	copy	idle	idle
8	idle	idle	OfficeTemp(28) — Tell(18,22, Employee(Alan))	r145	idle	copy
9	idle	idle	OfficeTemp(28) — Employee(Alan),	idle	idle	idle
10	idle	idle	OfficeTemp(28) — Employee(Alan), OfficeTemp(28), greaterThanOrEqual(28, 25)	r147	idle	idle
11	idle	idle	OfficeTemp(28) — Employee(Alan), OfficeTemp(28), greaterThanOrEqual(28, 25), hasOfficeTemp(28, Hot)	r146	idle	copy
12	idle	idle	idle	idle	OfficeAircon(11) —, Employee(0001), Tell 22, 23, hasOfficeTemp(28, Hot))	r152
13	idle	idle	idle	idle	OfficeAircon(11) — Employee(Alan), hasOfficeTemp(28, Hot)), hasAirconStatus(11,On)	r156

Chapter 6

Conclusion And Future Work

In this Chapter, we conclude our research thesis and provide insights into future work. While providing context-aware services on resource-bounded devices is still a new area to be explored. A very satisfying number of research works are progressing toward the same area. In the conclusion section, we try to summarise our thesis.

6.1 Conclusion

This thesis has familiarised us with a new vision relating to the context-aware systems on small devices in a highly decentralised environment using agents. Since context-aware systems tend to adapt to their environment and are used mainly on smart devices. Therefore, problems arise with the usage of such devices, especially in terms of memory, computation power, battery

power, as discussed in previous chapters. This thesis contributed towards overcoming such issues by providing an algorithm which is small enough and has low complexity to run on devices which has lower resources . Further, the introduction of preferences for rule filtration is a novel approach to reduce the size of the rule-base and ultimately increase the overall efficiency by limiting the number of inputs. Since this thesis has provided solutions which span over a few major areas in computer science, and it is vital to summarise them chapter wise briefly. Chapter 1 introduces the user with the research problem statements and main contributions. The contributions discuss the algorithm, preferences and its types along with rule generation methodologies that are used in the later chapters. We provided our aim and concrete steps on how do we achieve the main aim. The Chapter also details the methodology to accomplish objectives, that will contribute towards achieving the aim.

Chapter 2 familiarise the reader with the notion of context, context-aware, and context-aware systems. Context-aware is one part of the areas that are covered in the same Chapter. Different types of context-aware systems are discussed such as location-aware, energy-aware, among others, followed by its different application to give us a nice idea of its applications in real life. Further, its design and architectural models are briefed to provide an idea of how a context-aware system can be designed in different ways. Next, Modelling of context is discussed. Followed by ontologies introduction. Different reasoning techniques are explained to give a better understanding of how the reasoning process works. The last section describes the rule-based systems.

Since the rule-based system is composed of different parts (rule-base, rule-execution, rule-matching, conflict resolution and act) integrated to work as a single body. Each part is distinctly defined with great details. As the major portion of this thesis deal with the RBS, it is important to have a clear understanding of RBS and its internal working mechanism. Chapter 2, basically introduced context-aware systems, ontologies and rule-based system and their inter-operation to provide one seamless framework.

Chapter 3 details the related work from context-aware systems and rule-based systems. However, rule-based systems for mobile devices are explained separately and comparison is made between different rule-engines and framework using Table. The Table illustrates various aspects that are vital for a rule-engine to possess, in order to have it categorised as a rule-engine for resource-bounded devices. Since rule engines work on matching rules with facts, and a well-known algorithm for such operation is RETE. The RETE algorithm developed by Charles Forgy is explained along with its analysis and the problems associated with it. The reason to mention the algorithm here is so that we can devise and overcome the issues related to the RETE algorithm, which is still widely used in industry for rule-based systems.

Chapter 4 defines in detail our proposed platform for smart devices backed with arguments as to why we choose a specific platform. Full schemas of context acquisition are clearly stated. Since rule-based systems do not have any particular format, we devise a format to be used in our framework. We developed efficient algorithms for every single phase of the rule-based system

and also calculated its complexities both in term of time and space. The end of the Chapter defines different methods on how a rule can be generated, which includes a specialised plugin to translate ontology and web-based rule generator. This Chapter mainly discusses the thesis contribution, which deals with the execution of the system.

Chapter 5 provides a novel approach by offering preferences before the start of the system and during execution and using live preferences, reducing the number of processable rules and increasing system performance by reducing redundancy. Without preferences, unwanted rules are processed, which may never be fired. While with preferences enabled, all the rules have higher chances of being matched and executed. This Chapter also has very detailed case studies ranging from simple to very detailed case studies, it covers different scenarios and in different operation level. We detailed case studies that are already published in reputable conferences and journals, while creating new case studies with dry-run analysis and implementation. For almost every case study, we calculated the average reduction in rules, maximum and minimum reduction in rules. The nature of results shows that the correctness of the system and performance of the system is not compromised.

While considering the scope of the proposed system, it still needs many features to enhance its working mechanism further. It is not possible to cover all the aspects in the short period, therefore the future work section defines our plans for the framework.

6.2 Future Work

The current work makes an effort to provide a direction for research in future. In our research, we achieved the core goals and made the system work with satisfactory results. However, there is always room for improvement from logical, practical and hardware platforms. Below we discuss some of the possibilities to enhance the framework further.

6.2.1 Hardware Advancement

The framework has the potential to be expanded and further explored for the better use of humankind. Since this thesis focused mainly on the technology of Android to test the feasibility of the framework. Similarly, other technologies, which are available on small devices such as Arduino¹, Raspberrypie² and LEGO Mindstorms³ have great potential to make use of the framework. This platform not only use a variety of sensors in different ways, but any sensor can be added/removed, which gives us more control over the hardware side. The Lego Mindstorm mainly provides sensors embedded framework with more focus on the robotic field. In this way, a moveable, context-aware system can be developed. Such usage can significantly reduce the cost of hardware as a single robot can move different places and collect context, which has to be received by separate sensors otherwise. All of the systems mentioned have

¹<https://www.arduino.cc/>

²<https://www.raspberrypi.org/>

³<https://www.lego.com/en-us/mindstorms>

mature communication, processing and development environment with the availability of 3rd party sensors, which makes them a very nice options for context-aware devices.

6.2.2 Unified Ruling System

Another future progress can be on the rule generation strategy. Since the rules are designed in some software or using some tool and then processed traditionally. In our case, we have a variety of channels for generation and editing of the rules such as web-based interface, writing to JSON file or using Onto-HCR tool. Further, we make changes to the generated rule for preferences and likewise. It will be more convenient if dedicated research is carried out in this direction, to make one standalone system capable of doing all of the tasks as mentioned above and with automation.

6.2.3 Reversal Of Preference Set

Although, in our proposed system, we have a well-explained mechanism for deriving a preference set from a set of rules. This set depends on a variety of choices by a user such as context-based, derived or live preferences. However, it is unavoidable to make a strategy that can reverse a set of rules when not required. The preference set may be able to remove rules which are not likely to fire in the future by its own. By doing this, the redundancy will be more reduced to maximise the system output.

6.2.4 Working Memory Limitation And Updation

In this section, we discuss some possible conceptual solutions towards the working memory limitation. As the working memory has a limitation, the limitation has to follow some methodology. Below we discuss potential methods to explore further.

Distinct WM

In the database analogy, the distinct returns all the results so that the duplicated values are only shown once instead of repeating it. Similarly, the working memory maximum limit can be put equal to the number of distinct consequences of the rules. If there are a hundred rules and there are some rules which have the same consequences, then the working memory can be equal to the distinct size of total consequences. Mathematically Let R be the total number of rules, let R_c be the total consequences of the rules. Let RD_c be the distinct consequences such that $RD_c \leq R_c$ AND $RD_c > 0$, and size of $WM = RD_c$

Average of the preference sets

In this technique, the preferences sets are taken into consideration. It is more complicated than the previous method with a more space-saving mechanism when preferences are supposed to implement. This mechanism considers the rules in different preference sets, and takes the average of the set and create a working memory of the average size. Mathematically, Let R be a rule

base. Let P_1, P_2, P_3 be three different preference set with varied preference methods. Let R_x be a single rule instance, where $R_x \subset (P_1 \cup P_2 \cup P_3)$, Let P_a be the average of the preference sets, and n be the number of the preference sets available and n_r be the number of total rules in all preference sets we get $P_a = n_r/n$. The size of the working memory become equal to P_a .

User assigned

In this case, it is totally in the control of a user. A user can determine the size of the working memory and based on his/her knowledge of the rules, and any size can be provided as far as it is available. This particular procedure is the case presented in the thesis.

There is always a chance that a newer element in some cases can overwrite an element in working memory. How this overwritten is handled currently is randomised overwriting, which may lead to unwanted delays if not correct results. Therefore, this area can also be one of a complex research problem. It should be efficient enough to replace any context that has little to no impact on the overall results.

Further, some more realistic advancement can be proposed in the communication model, depending on the technology used and distances of the devices from each other. Variety of communication channels can be used. The mode of communication has to be energy efficient and reliable enough to send/receive data on time accurately.

Appendix A

Complete List Of Rules

In this section, a complete rules sets required to test different scenarios from a person daily routine are provided. The rules are arranged agent wise. The initial facts required to initiate any agent are also provided. Further, the preferences are added to understand which rules are added and which will be excluded in the runtime of the system where applicable. As the execution indicates, some of the rules are entirely excluded without affecting the output.

Table A.1: Set of rules for Smart Patient

Rule No	Agent Name and Facts	Condition	Consequent	Preference
1	Patient Care Agent (1)	PatientID(?pid), Person(?p), hasPatientID(?p, ?pid)	Patient(?p)	.
2	PatientID(01), Person(Alan), hasPatientID(Alan,01)	Patient(?p), hasFever(?p, Hyperpyrexia)	hasSituation(?p, Emergency)	Patient(Alan)
3		Patient(?p), hasBloodPressure(?p, High), hasDBCategory(?p, Low), hasFever(?p, High), hasPulseRate(?p, High)	hasSituation(?p, Emergency)	Patient(Alan)

Table A.1: Set of rules for Smart Patient

Rule No	Agent Name and Facts	Condition	Consequent	Preference
4		Patient(?p), hasBloodPressure(?p, Low), hasDBCategory(?p, Low), hasFever(?p, High), hasPulseRate(?p, Low)	hasSituation(?p, Emergency)	Patient(Alan)
5		Patient(?p), hasBloodPressure(?p, High), hasDBCategory(?p, High), hasFever(?p, High), hasPulseRate(?p, High)	hasSituation(?p, Emergency)	Patient(Alan)
6		Patient(?p), hasPulseRate(?p, Abnormal)	hasSituation(?p, Emergency)	Patient(Alan)
7		Patient(?p), hasDBCategory(?p, Normal)	hasNotSituation(?p, Emergency)	GPSLocation(Hospital)
8		Patient(?p), hasDBCategory(?p, Hypoglycaemia)	hasSituation(?p, Emergency)	Patient(Alan)
9		Patient(?p), hasFever(?p, Normal)	hasNotSituation(?p, Emergency)	GPSLocation(Hospital)
10		Patient(?p), hasBloodPressure(?p, Normal)	hasNotSituation(?p, Emergency)	GPSLocation(Hospital)
11		Patient(?p), hasBloodPressure(?p, Hypotention)	hasSituation(?p, Emergency)	Patient(Alan)
12		Patient(?p), hasPulseRate(?p, Normal)	hasNotSituation(?p, Emergency)	GPSLocation(Hospital)
13		Patient(?p), hasFever(?p, Hypopyrexia)	hasSituation(?p, Emergency)	Patient(Alan)
14		Patient(?p), hasBloodPressure(?p, Hypertention)	hasSituation(?p, Emergency)	Patient(Alan)
15		Patient(?p), hasDBCategory(?p, Hyperglycaemia)	hasSituation(?p, Emergency)	Patient(Alan)
16		Nurse(?nurse), Patient(?p), hasBloodPressure(?p, Low)	isCaredBy(?p, ?nurse)	GPSLocation(Hospital)

Table A.1: Set of rules for Smart Patient

Rule No	Agent Name and Facts	Condition	Consequent	Preference
17		Nurse(?nurse), Patient(?p), hasPulseRate(?p, High)	isCaredBy(?p, ?nurse)	GPSLocation(Hospital)
18		Nurse(?nurse), Patient(?p), has- Fever(?p, High)	isCaredBy(?p, ?nurse)	GPSLocation(Hospital)
19		Nurse(?nurse), Patient(?p), hasDB- Category(?p, High)	isCaredBy(?p, ?nurse)	GPSLocation(Hospital)
20		Nurse(?nurse), Patient(?p), hasBlood- Pressure(?p, High)	isCaredBy(?p, ?nurse)	GPSLocation(Hospital)
21		Nurse(?nurse), Patient(?p), hasPulseRate(?p, Low)	isCaredBy(?p, ?nurse)	GPSLocation(Hospital)
22		Caregiver(?caregiver), Patient(?p)	hasCaregiver(?p, ?caregiver)	GPSLocation(Hospital)
23		Nurse(?nurse), Patient(?p), hasDB- Category(?p, Low)	isCaredBy(?p, ?nurse)	GPSLocation(Hospital)
24		Tell(2, 1, hasBloodPressure(?p, Low))	hasBloodPressure(?p, Low)	Patient(Alan)
25		Tell(3, 1, hasDBCategory(?p, Hypo- glycaemia))	hasDBCategory(?p, Hypogly- caemia)	Patient(Alan)
26		Tell(3, 1, hasDBCategory(?p, Nor- mal))	hasDBCategory(?p, Normal)	Patient(Alan)
27		Tell(4, 1, hasFever(?p, Low))	hasFever(?p, Low)	Patient(Alan)
28		Tell(2, 1, hasBloodPressure(?p, Nor- mal))	hasBloodPressure(?p, Normal)	Patient(Alan)
29		Tell(3, 1, hasDBCategory(?p, High))	hasDBCategory(?p, High)	Patient(Alan)
30		Tell(4, 1, hasFever(?p, High))	hasFever(?p, High)	Patient(Alan)
31		Tell(2, 1, hasBloodPressure(?p, Hypo- tention))	hasBloodPressure(?p, Hypo- tention)	Patient(Alan)
32		Tell(3, 1, hasDBCategory(?p, Hyper- glycaemia))	hasDBCategory(?p, Hypergly- caemia)	Patient(Alan)
33		Tell(5, 1, hasPulseRate(?p, High))	hasPulseRate(?p, High)	Patient(Alan)
34		Tell(5, 1, hasPulseRate(?p, Low))	hasPulseRate(?p, Low)	Patient(Alan)
35		Tell(3, 1, hasDBCategory(?p, Low))	hasDBCategory(?p, Low)	Patient(Alan)

Table A.1: Set of rules for Smart Patient

Rule No	Agent Name and Facts	Condition	Consequent	Preference
36		Tell(2, 1, hasBloodPressure(?p, High))	hasBloodPressure(?p, High)	Patient(Alan)
37		Tell(4, 1, hasFever(?p, Hyperpyrexia))	hasFever(?p, Hyperpyrexia)	Patient(Alan)
38		Tell(2, 1, hasBloodPressure(?p, Hypertention))	hasBloodPressure(?p, Hypertention)	Patient(Alan)
39		Tell(4, 1, hasFever(?p, Normal))	hasFever(?p, Normal)	Patient(Alan)
40		Tell(5, 1, hasPulseRate(?p, Abnormal))	hasPulseRate(?p, Abnormal)	Patient(Alan)
41		Tell(5, 1, hasPulseRate(?p, Normal))	hasPulseRate(?p, Normal)	Patient(Alan)
42		Patient(?p), hasSituation(?p, Emergency)	Tell(1, 6, hasSituation(?p, Emergency))	Patient(Alan)
43	Blood Pressure Monitor (2)	DiastolicBP(?dbp), Person(?p), SystolicBP(?sbp), hasDiastolicBP(?p, ?dbp), hasSystolicBP(?p, ?sbp), lessThan(?dbp, 70), lessThan(?sbp, 110)	hasBloodPressure(?p, Low)	GPSLocation(Hospital)
44	DiastolicBP(90), Person(Alan), SystolicBP(130),hasDiastolicBP(Alan,90), hasSystolicBP(Alan, 130),greaterThanOrEqual(90, 85),greaterThanOrEqual(130, 125)	DiastolicBP(?dbp), Person(?p), SystolicBP(?sbp), hasDiastolicBP(?p, ?dbp), hasSystolicBP(?p, ?sbp), lessThan(?dbp, 60), lessThan(?sbp, 90)	hasBloodPressure(?p, Hypotention)	GPSLocation(Hospital)
45		DiastolicBP(?dbp), Person(?p), SystolicBP(?sbp), hasDiastolicBP(?p, ?dbp), hasSystolicBP(?p, ?sbp), greaterThanOrEqual(?dbp, 85), greaterThanOrEqual(?sbp, 125)	hasBloodPressure(?p, High)	.

Table A.1: Set of rules for Smart Patient

Rule No	Agent Name and Facts	Condition	Consequent	Preference
46		DiastolicBP(?dbp), Person(?p), SystolicBP(?sbp), hasDiastolicBP(?p, ?dbp), hasSystolicBP(?p, ?sbp), greaterThanOrEqualTo(?dbp, 95), greaterThanOrEqualTo(?sbp, 150)	hasBloodPressure(?p, Hypertention)	GPSLocation(Hospital)
47		DiastolicBP(?dbp), Person(?p), SystolicBP(?sbp), hasDiastolicBP(?p, ?dbp), hasSystolicBP(?p, ?sbp), greaterThanOrEqualTo(?dbp, 70), greaterThanOrEqualTo(?sbp, 110), lessThan(?dbs, 85), lessThan(?sbp, 125)	hasBloodPressure(?p, Normal)	GPSLocation(Hospital)
48		hasBloodPressure(?p, Hypotention)	Tell(2, 1, hasBloodPressure(?p, Hypotention))	GPSLocation(Hospital)
49		hasBloodPressure(?p, Normal)	Tell(2, 1, hasBloodPressure(?p, Normal))	GPSLocation(Hospital)
50		hasBloodPressure(?p, Hypertention)	Tell(2, 1, hasBloodPressure(?p, Hypertention))	GPSLocation(Hospital)
51		hasBloodPressure(?p, High)	Tell(2, 1, hasBloodPressure(?p, High))	hasBloodPressure(?p, High)
52	Diabetes Meter (3)	BloodSugarLevel(?bsl), Person(?p), lessThan(?bsl, 50)	hasDBCATEGORY(?p, Hypoglycaemia)	GPSLocation(Hospital)
53	BloodSugarLevel(250), Person(Alan), greaterThanOrEqualTo(250, 130), lessThan(250, 300)	BloodSugarLevel(?bsl), Person(?p), greaterThanOrEqualTo(?bsl, 300)	hasDBCATEGORY(?p, Hyperglycaemia)	GPSLocation(Hospital)
54		BloodSugarLevel(?bsl), Person(?p), lessThan(?bsl, 80), greaterThanOrEqualTo(?bsl, 50)	hasDBCATEGORY(?p, Low)	GPSLocation(Hospital)
55		BloodSugarLevel(?bsl), Person(?p), greaterThanOrEqualTo(?bsl, 80), lessThan(?bsl, 130)	hasDBCATEGORY(?p, Normal)	GPSLocation(Hospital)

Table A.1: Set of rules for Smart Patient

Rule No	Agent Name and Facts	Condition	Consequent	Preference
56		BloodSugarLevel(?bsl), Person(?p), greaterThanOrEqual(?bsl, 130), lessThan(?bsl, 300)	hasDBCategory(?p, High)	.
57		hasDBCategory(?p, Low)	Tell(3, 1, hasDBCategory(?p, Low))	GPSLocation(Hospital)
58		hasDBCategory(?p, Hyperglycaemia)	Tell(3, 1, hasDBCategory(?p, Hyperglycaemia))	GPSLocation(Hospital)
59		hasDBCategory(?p, Normal)	Tell(3, 1, hasDBCategory(?p, Normal))	GPSLocation(Hospital)
60		hasDBCategory(?p, High)	Tell(3, 1, hasDBCategory(?p, High))	hasDBCategory(?p, High)
61		hasDBCategory(?p, Hypoglycaemia)	Tell(3, 1, hasDBCategory(?p, Hypoglycaemia))	GPSLocation(Hospital)
62	Fever Monitor (4)	BodyTemperature(?temp), Per- son(?p), hasBodyTemperature(?p, ?temp), greaterThanOrEqual(?temp, 103)	hasFever(?p, Hyperpyrexia)	GPSLocation(Hospital)
63	BodyTemperature(102),person(Alan), gasBodyTemperature(Alan,102), greaterThanOrEqual(102, 99), lessThan(102, 103)	BodyTemperature(?temp), Per- son(?p), hasBodyTemperature(?p, ?temp), lessThan(?temp, 96)	hasFever(?p, Low)	GPSLocation(Hospital)
64		BodyTemperature(?temp), Per- son(?p), hasBodyTemperature(?p, ?temp), greaterThanOrEqual(?temp, 96), lessThan(?temp, 99)	hasFever(?p, Normal)	GPSLocation(Hospital)
65		BodyTemperature(?temp),person(?p), hasBodyTemperature(?p,?temp), greaterThanOrEqual(?temp, 99), lessThan(?temp, 103)	hasFever(?p, High)	.
66		hasFever(?p, Hyperpyrexia)	Tell(4, 1, hasFever(?p, Hyper- pyrexia))	GPSLocation(Hospital)

Table A.1: Set of rules for Smart Patient

Rule No	Agent Name and Facts	Condition	Consequent	Preference
67		hasFever(?p, High)	Tell(4, 1, hasFever(?p, High))	hasFever(Alan, High)
68		hasFever(?p, Normal)	Tell(4, 1, hasFever(?p, Normal))	GPSLocation(Hospital)
69		hasFever(?p, Low)	Tell(4, 1, hasFever(?p, Low))	GPSLocation(Hospital)
70	Pulse Monitor (5)	Person(?p), Pulse(?pulse), hasPulse(?p, ?pulse), greaterThanOrEqualTo(?pulse, 100), lessThan(?pulse, 120)	hasPulseRate(?p, High)	.
71	Person(Alan), Pulse(115), hasPulse(Alan, 115), greaterThanOrEqualTo(115, 100), lessThan(115, 120)	Person(?p), Pulse(?pulse), hasPulse(?p, ?pulse), greaterThanOrEqualTo(?pulse, 60), lessThan(?pulse, 100)	hasPulseRate(?p, Normal)	GPSLocation(Hospital)
72		Person(?p), Pulse(?pulse), hasPulse(?p, ?pulse), lessThan(?pulse, 50)	hasPulseRate(?p, Abnormal)	GPSLocation(Hospital)
73		hasPulseRate(?p, Abnormal)	Tell(5, 1, hasPulseRate(?p, Abnormal))	GPSLocation(Hospital)
74		hasPulseRate(?p, Normal)	Tell(5, 1, hasPulseRate(?p, Normal))	GPSLocation(Hospital)
75		Person(?p), Pulse(?pulse), hasPulse(?p, ?pulse), greaterThanOrEqualTo(?pulse, 120)	hasPulseRate(?p, Abnormal)	GPSLocation(Hospital)
76		Person(?p), Pulse(?pulse), hasPulse(?p, ?pulse), greaterThanOrEqualTo(?pulse, 50), lessThan(?pulse, 60)	hasPulseRate(?p, Low)	GPSLocation(Hospital)
77		hasPulseRate(?p, High)	Tell(5, 1, hasPulseRate(?p, High))	hasPulseRate(?p, High)
78		hasPulseRate(?p, Low)	Tell(5, 1, hasPulseRate(?p, Low))	GPSLocation(Hospital)

Table A.1: Set of rules for Smart Patient

Rule No	Agent Name and Facts	Condition	Consequent	Preference
79	Emergency Monitor (6)	Tell(1, 6, hasSituation(?p, Emergency))	hasSituation(?p, Emergency)	.
80	Patient(Alan),Physician(Alex)	Tell(7, 6, hasGPSLocation(?p, ?loc))	hasGPSLocation(?p, ?loc)	.
81		Ask (6, 7, hasGPSLocation(?p, ?loc))	hasGPSLocation(?p, ?loc)	.
82		Patient(?p), Physician(?physc), hasGPSLocation(?p, ?loc), hasSituation(?p, Emergency)	isDiagnosedBy(?p, ?physc)	.
83		Patient(?p), hasGPSLocation(?p, ?loc), hasSituation(?p, Emergency)	isRescuedBy(?p, ?amb)	.
84	GPS Sensor (7)	Ask(6,7, hasGPSLocation(?p, ?loc))	hasGPSLocation(?p, ?loc)	.
85	hasGPSLocation(Alan,UNMC)	hasGPSLocation(?p, ?loc)	Tell(7, 6, hasGPSLocation(?p, ?loc))	.

Table A.2: Set of rules for Smart Home

Rule No	Agent Name and facts	Condition	Consequent	Preference
86	AuthorizationSensor (8)	AuthorizationID(?aid), BiometricSensor(?bid), Person(?p), hasAuthorizationID(?p, ?aid), hasBiometricID(?p, ?bid)	isAuthorizedPerson(?p, Yes)	GPSLocation(home)
87	AuthorizationID(A01), BiometricSensor(B01), Person(Alan), hasAuthorizationID(Alan, A01), hasBiometricID(Alan, B01)	isAuthorizedPerson(?p, Yes)	Tell(8, 9, isAuthorizedPerson(?p, Yes))	GPSLocation(home)
88	.	isAuthorizedPerson(?p, Yes)	Tell(8, 15, isAuthorizedPerson(?p, Yes))	GPSLocation(home)
89	MotionDetector (9)	hasOccupancy(?p, Yes)	Tell(9, 10, hasOccupancy(?p, Yes))	GPSLocation(home)

Table A.2: Set of rules for Smart Home

Rule No	Agent Name and facts	Condition	Consequent	Preference
90	Room(Bedroom), MotionDetector(True), isAvailableAt(Alan, Bedroom)	hasOccupancy(?p, Yes)	Tell(9, 11, hasOccupancy(?p, Yes))	GPSLocation(home)
91	.	hasOccupancy(?p, Yes)	Tell(9, 12, hasOccupancy(?p, Yes))	GPSLocation(home)
92	.	MotionDetector(?true), isAuthorizedPerson(?p, Yes)	hasOccupancy(?p, Yes)	GPSLocation(home)
93	.	isAvailableAt(?p, ?room), Room(?room)	Tell(9, 10, isAvailableAt(?p, ?room))	GPSLocation(home)
94	.	Tell(8, 9, isAuthorizedPerson(?p, Yes))	isAuthorizedPerson(?p, Yes)	GPSLocation(home)
95	.	isAvailableAt(?p, ?room), Room(?room)	Tell(9, 11, isAvailableAt(?p, ?room))	GPSLocation(home)
96	LightSensor (10)	Tell(9, 10, isAvailableAt(?p, ?room))	isAvailableAt(?p, ?room)	isAuthorizedPerson(John, Yes)
97	.	isAvailableAt(?p, ?room), hasOccupancy(?p, No)	hasLightStatus(?p, Off)	isAuthorizedPerson(John, Yes)
98	.	Tell(9, 10, hasOccupancy(?p, Yes))	hasOccupancy(?p, Yes)	isAuthorizedPerson(John, Yes)
99	.	isAvailableAt(?p, ?room), hasOccupancy(?p, Yes)	hasLightStatus(?p, On)	isAuthorizedPerson(John, Yes)
100	Aircon Controller (11)	isAvailableAt(?p, ?room), hasOccupancy(?p, Yes), hasTemp(?t, Hot)	hasAirconStatus(?room, On)	GPSLocation(home)
101	.	isAvailableAt(?p, ?room), hasOccupancy(?p, Yes), hasTemp(?t, Cool)	hasAirconStatus(?room, Off)	GPSLocation(home)
102	.	Tell(9, 11, hasOccupancy(?p, Yes))	hasOccupancy(?p, Yes)	GPSLocation(home)
103	.	Tell(13, 11, hasTemp(?t, Cool))	hasTemp(?t, Cool)	GPSLocation(home)

Table A.2: Set of rules for Smart Home

Rule No	Agent Name and facts	Condition	Consequent	Preference
104	.	Tell(13, 11, hasTemp(?t, Hot))	hasTemp(?t, Hot)	GPSLocation(home)
105	.	Tell(9, 11, isAvailableAt(?p, ?room))	isAvailableAt(?p, ?room)	GPSLocation(home)
106	Home Controller Sensor (12)	Tell(9, 12, hasOccupancy(?p, Yes))	hasOccupancy(?p, Yes)	isAuthorizedPerson(John, Yes)
107	.	hasDoorStatus(?door, Open), hasOccupancy(?p, Yes)	hasDoorStatus(?door, Close)	isAuthorizedPerson(John, Yes)
108	.	hasDoorStatus(?door, Open), hasOccupancy(?p, No)	hasAlert(?p, OpenDoor)	isAuthorizedPerson(John, Yes)
109	.	Tell(14,12, hasDoorStatus(?door, Open))	hasDoorStatus(?door, Open)	isAuthorizedPerson(John, Yes)
110	Temperature Sensor (13)	Temperature(?t), lessThan(?t, 16)	hasTemp(?t, Cool)	GPSLocation(home)
111	Temperature(30)	hasTemp(?t, Cool)	Tell(13, 11, hasTemp(?t, Cool))	GPSLocation(home)
112	.	hasTemp(?t, Hot)	Tell(13, 11, hasTemp(?t, Hot))	GPSLocation(home)
113	.	Temperature(?t), greaterThanOrEqualTo(?t, 25)	hasTemp(?t, Hot)	GPSLocation(home)
114	.	Temperature(?t), greaterThanOrEqualTo(?t, 16), lessThan(?t, 25)	hasTemp(?t, Normal)	GPSLocation(home)
115	Door Control Sensor (14)	hasDoorStatus(?door, Open)	Tell(14,12, hasDoorStatus(?door, Open))	isAuthorizedPerson(John, Yes)
116	Gas Leak Detector (15)	LeakDetector(?ld), LeakDetected(?ld, True)	hasAlert(?p, GasLeak)	isAuthorizedPerson(John, Yes)
117	Smoke Sensor (16)	Smoke(?s), isSmokeDetected(?s, True)	hasAlert(?p, SmokeDetected)	isAuthorizedPerson(John, Yes)

Table A.3: Set of rules for Smart Office

Rule No	Agent Name and facts	Condition	Consequent	Preference
118	Smart Task Manager (17)	Employee(?p), TaskList(?task), TaskPending(?p, ?task)	isNotTaskDone(?p, Yes)	Employee(John)
119	.	Employee(?p), TaskList(?task), DesignatedTask(?p, ?task)	isTaskDone(?p, Yes)	Employee(John)
120	.	Tell(18, 17, Employee(?p))	Employee(?p)	Employee(John)
121	Employee (18)	EmployeeID(?empID), Person(?p), hasEmployeeID(?p, ?empID)	Employee(?p)	GPSLocation(UNMC)
122	EmployeeID(?empID), Person(?p)	Employee(?p)	Tell(18,17, Employee(?p))	Employee(Alan)
123	.	Employee(?p)	Tell(18,19, Employee(?p))	Employee(Alan)
124	.	Employee(?p)	Tell(18,20, Employee(?p))	Employee(Alan)
125	.	Employee(?p)	Tell(18,21, Employee(?p))	Employee(Alan)
126	.	Employee(?p)	Tell(18, 22, Employee(?p))	Employee(Alan)
127	.	Employee(?p)	Tell(18, 23, Employee(?p))	Employee(Alan)
128	Smart Chair (19)	Employee(?p), SmartChair(?sc)	hasSmartChair(?p, ?sc)	Employee(John)
129	.	Tell(18,19, Employee(?p))	Employee(?p)	Employee(John)
130	.	Posture(?pos), Time(?time), hasSmartChair(?p, ?sc), greaterThanOrEqual(?time, 15)	ChangePosture(?p, ?pos)	Employee(John)
131	Light Lamp (20)	hasLightLumens(?l, ?lumens), isGreaterThen(?lumens, 5000), isLessThen(?lumens, 10000)	hasLightStatus(?l, Normal)	Employee(John)
132	.	hasLightLumens(?l, ?lumens), isLessThen(?lumens, 5000)	hasLightStatus(?l, Dim)	Employee(John)

Table A.3: Set of rules for Smart Office

Rule No	Agent Name and facts	Condition	Consequent	Preference
133	.	hasLightLumens(?l, ?lumens), isGreaterThen(?lumens, 10000)	hasLightStatus(?l, Bright)	Employee(John)
134	.	Tell(18,20, Employee(?p))	Employee(?p)	Employee(John)
135	.	hasLightStatus(?l, Dim)	Tell(20, 21, hasLightStatus(?l, Dim)	Employee(John)
136	.	hasLightStatus(?l, Bright)	Tell(20, 21, hasLightStatus(?l, Bright)	Employee(John)
137	Window Blinds (21)	Blinds(?curtin), BlindsPosi- tion(?curtin, down), hasLight- Status(?l, Dim)	hasChangeBlindsStatus(?curtin, up)	Employee(John)
138	.	Tell(18,21, Employee(?p))	Employee(?p)	Employee(John)
139	.	Tell(20, 21, hasLightStatus(?l, Bright)	hasLightStatus(?l, Bright)	Employee(John)
140	.	Tell(20, 21, hasLightStatus(?l, Dim)	hasLightStatus(?l, Dim)	Employee(John)
141	.	Blinds(?curtin), BlindsPosi- tion(?curtin, up), hasLight- Status(?l, Bright)	hasChangeBlindsStatus(?curtin, down)	Employee(John)
142	.	Blinds(?curtin), hashasBlind- Status(?curtin, ?status)	BlindsPosition(?curtin, down)	Employee(John)
143	.	Blinds(?curtin), hashasBlind- Status(?curtin, ?status)	BlindsPosition(?curtin, up)	Employee(John)
144	Office Temperature (22)	OfficeTemp(?temp), lessThanOrEqual(?temp, 16)	hasOfficeTemp(?temp, Cool)	Employee(Alan)
145	OfficeTemp(28)	Tell(18,22, Employee(?p))	Employee(?p)	Employee(Alan)
146	.	hasOfficeTemp(?temp, Hot) , Employee(?p)	Tell(22, 23, hasOf- ficeTemp(?temp, Hot))	Employee(Alan)

Table A.3: Set of rules for Smart Office

Rule No	Agent Name and facts	Condition	Consequent	Preference
147	.	OfficeTemp(?temp), greaterThanOrEqualTo(?temp, 25)	hasOfficeTemp(?temp, Hot)	Employee(Alan)
148	.	OfficeTemp(?temp), greaterThanOrEqualTo(?temp, 16), lessThan(?temp, 25)	hasOfficeTemp(?temp, Normal)	Employee(Alan)
149	.	hasOfficeTemp(?temp, Cool) , Employee(?p)	Tell(22, 23, hasOfficeTemp(?temp, Cool))	Employee(Alan)
150	Office Aircon (23)	OfficeAircon(?ac), hasOfficeTemp(?temp, Hot) , Employee(?P)	hasAirconStatus(?ac, On)	Employee(Alan)
151	OfficeAircon(11)	OfficeAircon(?ac), hasOfficeTemp(?temp, Cool) ,Employee(?P)	hasAirconStatus(?ac, Off)	Employee(Alan)
152	.	Tell(22, 23, hasOfficeTemp(?temp, Hot))	hasOfficeTemp(?temp, Hot)	Employee(Alan)
153	.	Tell(22, 23, hasOfficeTemp(?temp, Cool))	hasOfficeTemp(?temp, Cool)	Employee(Alan)

Table A.4: Set of rules for Smart Shopping Cart

Rule No	Agent Name and facts	Condition	Consequent	Preference
154	Smart Shopping Cart (24)	BuyButterCube(?cube),Person(?p)	hasAlert(?p, BuyButter)	
155	Person(Alan)	BuyMilkPack(?pack),Person(?p)	hasAlert(?p, BuyMilk)	
156	.	BuyEgg(?qty),Person(?p)	hasAlert(?p, BuyEgg)	
157	.	Tell(27, 24, BuyButterCube(?bqty))	BuyButterCube(?cube)	
158	.	Tell(26, 24, BuyMilkPack(?pack))	BuyMilkPack(?pack)	
159	.	Tell(25, 24, BuyEgg(?qty))	BuyEgg(?qty)	

Table A.5: Set of rules for Smart Shopping Cart

Rule No	Agent Name and facts	Condition	Consequent	Preference
160	Egg container sensors (25)	EggShelf(?qty), lessThanOrEqual(?qty, 4)	BuyEgg(?qty)	
161	.	BuyEgg(?qty)	Tell(25, 24, BuyEgg(?qty))	
162	Milk Pack Quantity Sensor (26)	MilkPack(?pack), lessThanOrEqual(?pack, 2)	BuyMilk(?pack)	
163	.	BuyMilk(?pack)	Tell(26, 24, BuyMilk-Pack(?pack))	
164	Butter Cube Quantity Sensor (27)	Bucket(?bqty), , lessThanOrEqual(?bqty, 4)	BuyButter(?bqty)	
165	.	BuyButter(?bqty)	Tell(27, 24, BuyButter-Cube(?bqty))	
166	Fridge Door Sensor (28)	FridgeDoor(?door), SensorStatus(Open)	hasStatus(?door, Open)	.
167	.	hasStatus(?door, Open)	hasAlert(?p, DoorOpen)	.

Bibliography

- [1] M. INC, “Motorola demonstrates portable telephone to be available for public use by 1976,” April 3 1973. Press Release from Motorola Inc.
- [2] R. Ballagas, J. Borchers, M. Rohs, and J. G. Sheridan, “The smart phone: a ubiquitous input device,” *Pervasive Computing, IEEE*, vol. 5, no. 1, pp. 70–77, 2006.
- [3] S. Schrittwieser, P. Frühwirt, P. Kieseberg, M. Leithner, M. Mulazzani, M. Huber, and E. R. Weippl, “Guess who’s texting you? evaluating the security of smartphone messaging applications,” in *19th Annual Network and Distributed System Security Symposium*, 2012.
- [4] M. Weiser, “The computer for the 21st century,” *Scientific american*, vol. 265, no. 3, pp. 94–104, 1991.
- [5] C. Pei, H. Guo, X. Yang, Y. Wang, X. Zhang, and H. Ye, “Sensors in smart phone,” in *Computer and Computing Technologies in Agriculture IV*, pp. 491–495, Springer, 2011.

- [6] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen, “Contextphone: A prototyping platform for context-aware mobile applications,” *Pervasive Computing, IEEE*, vol. 4, no. 2, pp. 51–59, 2005.
- [7] W. V. Woensel, N. A. Haider, P. C. Roy, A. M. Ahmad, and S. S. Abidi, “A comparison of mobile rule engines for reasoning on semantic web based health data,” in *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 01*, pp. 126–133, IEEE Computer Society, 2014.
- [8] G. J. Nalepa and B. Szymon, “Rule-based solution for context-aware reasoning on mobile devices,” *Computer Science and Information Systems*, vol. 11, no. 1, pp. 171–193, 2014.
- [9] C. Mukherjee, *Which Rules Engine Is Best for Building Smart Applications?*, pp. 3–14. Berkeley, CA: Apress, 2018.
- [10] B. N. Schilit and M. M. Theimer, “Disseminating active map information to mobile hosts,” *Network, IEEE*, vol. 8, no. 5, pp. 22–32, 1994.
- [11] B. Schilit, N. Adams, and R. Want, “Context-aware computing applications,” in *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pp. 85–90, IEEE, 1994.

- [12] J. Pascoe, “Adding generic contextual capabilities to wearable computers,” in *Wearable Computers, 1998. Digest of Papers. Second International Symposium on*, pp. 92–99, IEEE, 1998.
- [13] A. K. Dey, “Understanding and using context,” *Personal and ubiquitous computing*, vol. 5, no. 1, pp. 4–7, 2001.
- [14] T. Winograd, “Architectures for context,” *Human-Computer Interaction*, vol. 16, no. 2, pp. 401–419, 2001.
- [15] A. Soylyu, P. De Causmaecker, and P. Desmet, “Context and adaptivity in context-aware pervasive computing environments,” in *Ubiquitous, Autonomic and Trusted Computing, 2009. UIC-ATC’09. Symposia and Workshops on*, pp. 94–101, IEEE, 2009.
- [16] H. Chen and S. Tolia, “Steps towards creating a context-aware software agent system,” *HP. Technical Report HPL-2001-231*, 2001.
- [17] P. E. Agre, “Changing places: contexts of awareness in computing,” *Human-computer interaction*, vol. 16, no. 2, pp. 177–192, 2001.
- [18] A. Schmidt, M. Beigl, and H.-W. Gellersen, “There is more to context than location,” *Computers & Graphics*, vol. 23, no. 6, pp. 893–901, 1999.
- [19] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, “The cricket location-support system,” in *Proceedings of the 6th annual interna-*

- tional conference on Mobile computing and networking*, pp. 32–43, ACM, 2000.
- [20] P. Castro and R. Muntz, “Using context to assist in multimedia object retrieval,” in *First International Workshop on Multimedia Intelligent Storage and Retrieval Management*, 1999.
- [21] T. Kindberg and J. Barton, “A web-based nomadic computing system,” *Computer Networks*, vol. 35, no. 4, pp. 443–456, 2001.
- [22] B. Schilit, N. Adams, and R. Want, “Context-aware computing applications,” in *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pp. 85–90, IEEE, 1994.
- [23] S. Sehic, S. Nastic, M. Vögler, F. Li, and S. Dustdar, “Entity-adaptation: a programming model for development of context-aware applications,” in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pp. 436–443, ACM, 2014.
- [24] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, “Context aware computing for the internet of things: A survey,” *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 414–454, 2014.
- [25] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, “Towards a better understanding of context and context-awareness,” in *Handheld and ubiquitous computing*, pp. 304–307, Springer, 1999.

- [26] R. Want, A. Hopper, V. Falcão, and J. Gibbons, “The active badge location system,” *ACM Trans. Inf. Syst.*, vol. 10, no. 1, pp. 91–102, 1992.
- [27] D. Salber, A. K. Dey, and G. D. Abowd, “The context toolkit: Aiding the development of context-enabled applications,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, (New York, NY, USA), pp. 434–441, ACM, 1999.
- [28] J. E. Bardram, , and N. Nørskov, “A context-aware patient safety system for the operating room,” in *Proceedings of the 10th international conference on Ubiquitous computing*, pp. 272–281, 2008.
- [29] M. Wooldridge, *An Introduction to MultiAgent Systems*. Wiley Publishing, 2nd ed., 2009.
- [30] A. K. Dey, G. D. Abowd, and D. Salber, “A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications,” *Human-computer interaction*, vol. 16, no. 2, pp. 97–166, 2001.
- [31] S. Russell and P. Norvig, “Artificial intelligence: a modern approach,” 1995.
- [32] D. C. Smith, A. Cypher, and J. Spohrer, “Kidsim: programming agents without a programming language,” *Communications of the ACM*, vol. 37, no. 7, pp. 54–67, 1994.

- [33] M. Wooldridge and N. R. Jennings, “Agent theories, architectures, and languages: a survey,” in *International Workshop on Agent Theories, Architectures, and Languages*, pp. 1–39, Springer, 1994.
- [34] N. Petteri and F. Patrik, “Reasoning in context-aware systems. position paper,” 2004.
- [35] B. A. Michael, M. L. Sue, and C. Anthony, “Location-aware computing,” 2008.
- [36] J. L. Hernández, M. V. Moreno, A. J. Jara, and A. F. Skarmeta, “A soft computing based location-aware access control for smart buildings,” *Soft Computing*, vol. 18, no. 9, pp. 1659–1674, 2014.
- [37] B. Chakraborty and T. Hashimoto, “A framework for user aware route selection in pedestrian navigation system,” in *Aware Computing (ISAC), 2010 2nd International Symposium on*, pp. 150–153, IEEE, 2010.
- [38] D. Kliazovich, P. Bouvry, and S. U. Khan, “Greencloud: a packet-level simulator of energy-aware cloud computing data centers,” *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, 2012.
- [39] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein, “Dynamic energy-aware capacity provisioning for cloud computing environments,” in *Proceedings of the 9th international conference on Autonomic computing*, pp. 145–154, ACM, 2012.

- [40] R. Raju, J. Amudhavel, N. Kannan, and M. Monisha, “Interpretation and evaluation of various hybrid energy aware technologies in cloud computing environmenta detailed survey,” in *Green Computing Communication and Electrical Engineering (ICGCCEE), 2014 International Conference on*, pp. 1–3, IEEE, 2014.
- [41] A. Acharya, M. Ranganathan, and J. Saltz, “Sumatra: A language for resource-aware mobile programs,” in *Mobile Object Systems Towards the Programmable Internet*, pp. 111–130, Springer, 1997.
- [42] D. Molyneaux, S. Izadi, D. Kim, O. Hilliges, S. Hodges, X. Cao, A. Butler, and H. Gellersen, “Interactive environment-aware handheld projectors for pervasive computing spaces,” in *Pervasive Computing*, pp. 197–215, Springer, 2012.
- [43] A. E. Hassan, D. Chiu, and J. S. Wilson, “Computing device with environment aware features,” July 15 2008. US Patent 7,400,878.
- [44] R. Sriram, S. Geetha, J. Madhusudanan, P. Iyappan, V. P. Venkatesan, and M. Ganesan, “A study on context-aware computing framework in pervasive healthcare,” in *Proceedings of the 2015 International Conference on Advanced Research in Computer Science Engineering & Technology (ICARCSET 2015)*, p. 39, ACM, 2015.

- [45] R. J. Robles and T.-h. Kim, “Review: context aware tools for smart home development,” *International Journal of Smart Home*, vol. 4, no. 1, 2010.
- [46] E. S. Chan, D. Gawlick, A. Ghoneimy, and Z. H. Liu, “Situation aware computing for big data,” in *Big Data (Big Data), 2014 IEEE International Conference on*, pp. 1–6, IEEE, 2014.
- [47] P. J. Brown, J. D. Bovey, and X. Chen, “Context-aware applications: from the laboratory to the marketplace,” *Personal Communications, IEEE*, vol. 4, no. 5, pp. 58–64, 1997.
- [48] R. Nitesh, “Geo tagging and automatic generation of metadata for photos and videos,” May 19 2015. US Patent 9,037,583.
- [49] J. E. Bardram, “The java context awareness framework (jcaf)—a service infrastructure and programming framework for context-aware applications,” in *Pervasive computing*, pp. 98–115, Springer, 2005.
- [50] M. Böhmer, L. Ganev, and A. Krüger, “Appfunnel: A framework for usage-centric evaluation of recommender systems that suggest mobile applications,” in *Proceedings of the 2013 international conference on Intelligent user interfaces*, pp. 267–276, ACM, 2013.
- [51] A. Karatzoglou, L. Baltrunas, K. Church, and M. Böhmer, “Climbing the app wall: enabling mobile app discovery through context-aware

- recommendations,” in *Proceedings of the 21st ACM international conference on Information and knowledge management*, pp. 2527–2530, ACM, 2012.
- [52] M. Böhmer, G. Bauer, and A. Krüger, “Exploring the design space of context-aware recommender systems that suggest mobile applications,” in *2nd Workshop on Context-Aware Recommender Systems*, 2010.
- [53] W.-H. Rho and S.-B. Cho, “Context-aware smartphone application category recommender system with modularized bayesian networks,” in *Natural Computation (ICNC), 2014 10th International Conference on*, pp. 775–779, IEEE, 2014.
- [54] J. E. Bardram, “Applications of context-aware computing in hospital work: examples and design principles,” in *Proceedings of the 2004 ACM symposium on Applied computing*, pp. 1574–1579, ACM, 2004.
- [55] F. Ongenaë, P. Duysburgh, M. Verstraete, N. Sulmon, L. Bleumers, A. Jacobs, A. Ackaert, S. De Zutter, S. Verstichel, and F. De Turck, “User-driven design of a context-aware application: an ambient-intelligent nurse call system,” in *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2012 6th International Conference on*, pp. 205–210, IEEE, 2012.

- [56] G. W. Musumba and H. O. Nyongesa, “Context awareness in mobile computing: A review,” *International Journal of Machine Learning and Applications*, vol. 2, no. 1, pp. 5–pages, 2013.
- [57] Z. Rok, H. Marjan, and R. Ivan, “Taxonomy of context-aware systems,” *ELEKTROTEHNIKI VESTNIK*, vol. 79, no. 1-2, pp. 45–46, 2012.
- [58] S. Lee, J. Chang, and S.-g. Lee, “Survey and trend analysis of context-aware systems,” *Information-An International Interdisciplinary Journal*, vol. 14, no. 2, pp. 527–548, 2011.
- [59] A. M. Langer, *Guide to Software Development: Designing and Managing the Life Cycle*. Springer Science & Business Media, 2012.
- [60] G. Biegel and V. Cahill, “A framework for developing mobile, context-aware applications,” in *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pp. 361–365, IEEE, 2004.
- [61] T. Strang and C. Linnhoff-Popien, “A context modeling survey,” in *In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, 2004.
- [62] N. Guarino, D. Oberle, and S. Staab, “What is an ontology?,” in *Handbook on ontologies*, pp. 1–17, Springer, 2009.

- [63] T. R. Gruber, “A translation approach to portable ontology specifications,” *Knowledge acquisition*, vol. 5, no. 2, pp. 199–220, 1993.
- [64] T. R. Gruber, “Toward principles for the design of ontologies used for knowledge sharing?,” *International journal of human-computer studies*, vol. 43, no. 5, pp. 907–928, 1995.
- [65] N. F. Noy, D. L. McGuinness, *et al.*, “Ontology development 101: A guide to creating your first ontology,” 2001.
- [66] A. Rakib and H. M. U. Haque, “A logical framework for the representation and verification of context-aware agents,” *Mobile Networks and Applications*, vol. 19, no. 5, pp. 585–597, 2014.
- [67] A. Rakib, R. U. Faruqui, and W. MacCaull, “Verifying resource requirements for ontology-driven rule-based agents,” in *Foundations of Information and Knowledge Systems*, pp. 312–331, Springer, 2012.
- [68] J. Davies, R. Studer, and P. Warren, *Semantic Web technologies: trends and research in ontology-based systems*. John Wiley & Sons, 2006.
- [69] H. Boley, S. Tabet, and G. Wagner, “Design rationale for ruleml: A markup language for semantic web rules.,” in *SWWS*, vol. 1, pp. 381–401, 2001.

- [70] K. Latif, “Hybrid systems knowledge representation using modelling environment system techniques artificial intelligence,” *arXiv preprint arXiv:1409.1170*, 2014.
- [71] E. Lagun, “Evaluation and implementation of match algorithms for rule-based multi-agent systems using the example of jadex,”
- [72] J. C. Giarratano and G. Riley, “Expert systems, principles and programming, thomson course of technology,” *Boston, Australia*, 2005.
- [73] G. F. Luger, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving—6th Edition*.
- [74] E. Friedman, *Jess in action: rule-based systems in java*. Manning Publications Co., 2003.
- [75] A. Ligeza, *logical foundations of Rule-Based Systems*. Springer, 2006.
- [76] A. Gupta, *Parallelism in production systems*. Morgan Kaufmann, 1987.
- [77] G. Sehic, F. Li, S. Nastic, and S. Dustdar, “A programming model for context-aware applications in large-scale pervasive systems,” in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on*, pp. 142–149, IEEE, 2012.

- [78] M. Appeltauer, R. Hirschfeld, M. Haupt, and H. Masuhara, “Contextj: Context-oriented programming with java,” *Information and Media Technologies*, vol. 6, no. 2, pp. 399–419, 2011.
- [79] R. Hirschfeld, P. Costanza, and O. Nierstrasz, “Context-oriented programming,” *Journal of Object Technology*, vol. 7, no. 3, 2008.
- [80] B. Guo, D. Zhang, and M. Imai, “Toward a cooperative programming framework for context-aware applications,” *Personal and ubiquitous computing*, vol. 15, no. 3, pp. 221–233, 2011.
- [81] N. P. Lopes, J. A. Navarro, A. Rybalchenko, and A. Singh, “Applying prolog to develop distributed systems,” *Theory and Practice of Logic Programming*, vol. 10, no. 4-6, pp. 691–707, 2010.
- [82] D. Petcu and M. Petcu, “Distributed jess on a condor pool,” in *Proceedings of the 9th WSEAS International Conference on Computers*, p. 11, World Scientific and Engineering Academy and Society (WSEAS), 2005.
- [83] I. Chronis, A. Madan, and A. S. Pentland, “Socialcircuits: the art of using mobile phones for modeling personal interactions,” in *Proceedings of the ICMI-MLMI’09 Workshop on Multimodal Sensor-Based Systems and Mobile Phones for Social Computing*, p. 1, ACM, 2009.

- [84] J. J. Jung, “Contextualized mobile recommendation service based on interactive social network discovered from mobile users,” *Expert Systems with Applications*, vol. 36, no. 9, pp. 11950–11956, 2009.
- [85] D. O. Olguín, B. N. Waber, T. Kim, A. Mohan, K. Ara, and A. Pentland, “Sensible organizations: Technology and methodology for automatically measuring organizational behavior,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 1, pp. 43–55, 2009.
- [86] N. Eagle and A. S. Pentland, “Reality mining: sensing complex social systems,” *Personal and ubiquitous computing*, vol. 10, no. 4, pp. 255–268, 2006.
- [87] W. M. Aly, K. A. Eskaf, and A. S. Selim, “Fuzzy mobile expert system for academic advising,” in *Electrical and Computer Engineering (CCECE), 2017 IEEE 30th Canadian Conference on*, pp. 1–5, IEEE, 2017.
- [88] F. Sartori, L. Manenti, and L. Grazioli, “A conceptual and computational model for knowledge-based agents in android.,” *WOA@ AI* IA*, vol. 2013, pp. 41–46, 2013.
- [89] M. F. Abulkhair and L. F. Ibrahim, “Using rule base system in mobile platform to build alert system for evacuation and guidance,” *INTER-*

NATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, vol. 7, no. 4, pp. 68–79, 2016.

- [90] C. Mukherjee, *Build Android-Based Smart Applications: Using Rules Engines, NLP and Automation Frameworks*. Apress, 2017.
- [91] C. Mukherjee, *Issues Faced While Porting Rules Engines*, pp. 51–53. Berkeley, CA: Apress, 2018.
- [92] T. Gu, H. K. Pung, and D. Q. Zhang, “A middleware for building context-aware mobile services,” in *Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th*, vol. 5, pp. 2656–2660, IEEE, 2004.
- [93] H. Chen, *An intelligent broker architecture for pervasive context-aware systems*. PhD thesis, University of Maryland, Baltimore County, 2004.
- [94] H. M. Ul-Haque, *A formal approach to modelling and verification of context-aware systems*. PhD thesis, University of Nottingham, 2017.
- [95] C. L. Forgy, “Rete: A fast algorithm for the many pattern/many object pattern match problem,” *Artificial intelligence*, vol. 19, no. 1, pp. 17–37, 1982.
- [96] M. Slazynski, S. Bobek, and G. J. Nalepa, “Migration of rule inference engine to mobile platform. challenges and case study.,” *Knowledge Engineering and Software Engineering (KESE10)*, p. 71, 2014.

- [97] I. Uddin, H. M. U. Haque, A. Rakib, and M. R. S. Rahmat, “Resource-bounded context-aware applications: A survey and early experiment,” in *International Conference on Nature of Computation and Communication*, pp. 153–164, Springer, 2016.
- [98] R. B. Doorenbos, “Production matching for large learning systems.,” tech. rep., CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1995.
- [99] D. P. Miranker, D. A. Brant, B. J. Lofaso, and D. Gadbois, “On the performance of lazy matching in production systems.,” in *AAAI*, vol. 90, pp. 685–692, 1990.
- [100] D. Armstrong, *Memory Efficient Stream Reasoning on Resource-Limited Devices*. PhD thesis, Trinity College, 2014.
- [101] G. Liu, S. Huang, D. Zhang, and Y. Du, “A rete rule reasoning algorithm based on the audit method ontology,” *Int. J. Hybrid Inf. Technol*, vol. 7, pp. 211–244, 2014.
- [102] C. L. Forgy, *On the efficient implementation of production systems*. PhD thesis, Carnegie-Mellon University, 1979.
- [103] J. A. Kang and A. M. K. Cheng, “Shortening matching time in ops5 production systems,” *IEEE Transactions on Software Engineering*, vol. 30, no. 7, pp. 448–457, 2004.

- [104] Partho, “Top 10 java business rule engines,” 2009.
- [105] G. Developers, “Android studio,” 2015.
- [106] B. DeLacey, “Google calling, inside android, the gphone sdk,” 2007.
- [107] A. Rakib and H. M. U. Haque, “A logic for context-aware non-monotonic reasoning agents,” in *Human-Inspired Computing and Its Applications*, pp. 453–471, Springer, 2014.
- [108] R. G. Smith and R. Davis, “Frameworks for cooperation in distributed problem solving,” *IEEE Transactions on systems, man, and cybernetics*, vol. 11, no. 1, pp. 61–70, 1981.
- [109] A. Rakib and H. M. U. Haque, “A logic for context-aware non-monotonic reasoning agents,” in *Human-Inspired Computing and Its Applications*, pp. 453–471, Springer, 2014.
- [110] I. Uddin, A. Rakib, H. M. U. Haque, and P. C. Vinh, “Modeling and reasoning about preference-based context-aware agents over heterogeneous knowledge sources,” *Mobile Networks and Applications*, Sep 2017.
- [111] A. Rakib and I. Uddin, “An efficient rule-based distributed reasoning framework for resource-bounded systems,” *Mobile Networks and Applications*, vol. 24, no. 1, pp. 82–99, 2019.

- [112] F. Brandt, G. Chabin, and C. Geist, “Pnyx: : A powerful and user-friendly tool for preference aggregation,” in *AAMAS*, 2015.
- [113] A. Abbas, L. Zhang, and S. U. Khan, “A survey on context-aware recommender systems based on computational intelligence techniques,” *Computing*, vol. 97, no. 7, pp. 667–690, 2015.
- [114] J. Auda, D. Weber, A. Voit, and S. Schneegass, “Understanding user preferences towards rule-based notification deferral,” in *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, p. LBW584, ACM, 2018.
- [115] A. Mehrotra, R. Hendley, and M. Musolesi, “Prefminer: mining user’s preferences for intelligent mobile notification management,” in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 1223–1234, ACM, 2016.
- [116] C. Loitsch, G. Weber, N. Kaklanis, K. Votis, and D. Tzovaras, “A knowledge-based approach to user interface adaptation from preferences and for special needs,” *User Modeling and User-Adapted Interaction*, vol. 27, pp. 445–491, Dec 2017.
- [117] P. T. Moore and H. V. Pham, “Personalization and rule strategies in data-intensive intelligent context-aware systems,” *The Knowledge Engineering Review*, vol. 30, no. 2, pp. 140–156, 2015.

- [118] J. Manotumruksa, C. Macdonald, and I. Ounis, “Modelling user preferences using word embeddings for context-aware venue recommendation,” *arXiv preprint arXiv:1606.07828*, 2016.
- [119] M. F. Alhamid, M. Rawashdeh, H. Dong, M. A. Hossain, and A. El Sadik, “Exploring latent preferences for context-aware personalized recommendation systems,” *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 4, pp. 615–623, 2016.
- [120] H. Zhu, E. Chen, H. Xiong, K. Yu, H. Cao, and J. Tian, “Mining mobile user preferences for personalized context-aware recommendation,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 4, p. 58, 2015.
- [121] I. Uddin and A. Rakib, “A preference-based application framework for resource-bounded context-aware agents,” in *International Conference on Mobile and Wireless Technology*, pp. 187–196, Springer, 2017.
- [122] U. Ijaz and A. Rakib, “A resource-aware preference model for context-aware systems,” *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, 2017.