ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ


# ΚΑΤΑΝΕΜΗΜΕΝΟ ΣΥΣΤΗΜΑ ΑΝΑΙΡΕΣΙΜΗΣ ΣΥΛΛΟΓΙΣΤΙΚΗΣ ΓΙΑ ΚΙΝΗΤΕΣ ΣΥΣΚΕΥΕΣ ΣΕ ΠΕΡΙΒΑΛΛΟΝΤΑ ΔΙΑΧΥΤΗΣ ΝΟΗΜΟΣΥΝΗΣ

**Κωνσταντίνος Παπαθεοδώρου**

Μεταπτυχιακή Εργασία

Ηρακλειο, Δεκέμβριος 2010

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

# Κατανεμημένο Σύστημα Αναιρέσιμης Συλλογιστικής για Κινητές Συσκευές σε Περιβάλλοντα Διάχυτης Νοημοσύνης

Εργασία που υποβλήθηκε απο τον
**Κωνσταντίνο Παπαθεοδώρου**
ως μερική εκπλήρωση των απαιτήσεων για απόκτηση
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

_____
Κωνσταντίνος Παπαθεοδώρου, Πανεπιστήμιο Κρήτης

Εισηγητική Επιτροπή:

_____
Γρηγόρης Αντωνίου, Καθηγητής,
Πανεπιστήμιο Κρήτης, Επόπτης

_____
Δημήτρης Πλεξουσάκης, Καθηγητής,
Πανεπιστήμιο Κρήτης, Μέλος

_____
Γιάννης Τζίτζικας, Επίκουρος Καθηγητής,
Πανεπιστήμιο Κρήτης, Μέλος

Δεκτή από:

_____
Άγγελος Μπίλας, Αναπληρωτής Καθηγητής, Πανεπιστήμιο Κρήτης

Πρόεδρος επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Δεκέμβριος 2010

# A Distributed Defeasible Reasoning System for Mobile Devices in Ambient Intelligence Environments

Papatheodorou Constantinos

Master of Science Thesis

Computer Science Department, University of Crete

## Abstract

Ambient Intelligence environments consist of various devices that collect, process, change and share the available context information. The imperfect nature of context, the open and dynamic nature of ambient environments, and the special characteristics of the involved devices have introduced new research challenges on how to represent and reason with contextual information. In order to address such challenges, Bikakis et al. recently proposed *Contextual Defeasible Logic* (*CDL*), a formal framework for distributed contextual reasoning, integrating ideas from Defeasible Reasoning and Multi-Context Systems. CDL deals with ambiguities that may arise from the interaction of context theories, using defeasible mapping rules and employing a *per-context* preference relation represented as a total ordering on the system contexts.

Here, we extend this approach to additionally deal with incomplete preference information. To enable this, we replace total preference ordering with partial ordering, and modify the argumentation framework and the distributed algorithms that were originally proposed to meet the new requirements. We also describe the architecture of an implementation on small devices, and present the definition and implementation of two concrete application scenarios from the Ambient Intelligence and the Social Mobile Networks domains. Finally, we report on our initial experiences with the deployment of contextual defeasible reasoning on actual devices and our experimentation in real environments, and discuss performance and scalability issues of the proposed approach.

# Κατανεμημένο Σύστημα Αναιρέσιμης Συλλογιστικής για Κινητές Συσκευές σε Περιβάλλοντα Διάχυτης Νοημοσύνης

Παπαθεοδώρου Κωνσταντίνος

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

## Περίληψη

Τα περιβάλλοντα Διάχυτης Νοημοσύνης αποτελούνται απο διάφορες συσκευές που συλλέγουν, επεξεργάονται, μεταποιούν και διαμοιράζουν την διαθέσιμη πληροφορία περιβάλλοντος. Η ατελής γνώση για την πληροφορία περιβάλλοντος, η ανοικτή και δυναμική φύση των περιβάλλοντων Διάχυτης Νοημοσύνης και τα ειδικά χαρακτιριστικά των εμπλεκόμενων συσκευών έχουν εισάγει νέες προκλήσεις για έρευνα στον τομέα της αναπαράστασης και επεξεργασίας της πληροφορίας περιβάλλοντος. Για να αντιμετωπίσει αυτά τα προβλήματα, πρόσφατη εργασία (Bikakis et al.) πρότεινε ένα τυπικά ορισμένο πλαίσιο για κατανεμημένη συλλογιστική πάνω σε πληροφορία περιβάλλοντος, τη Αναιρέσιμη Συλλογιστική Περιβάλλοντος (Contextual Defeasible Logic ή CDL), όπου ενσωματώνονται ιδέες από την Αναιρέσιμη Συλλογιστική και τα Συστήματα Πολλαπλών Περιβαλλόντων (Multi-Context Systems). Η Αναιρέσιμη Συλλογιστική Περιβάλλοντος χειρίζεται τις ασυνέπειες που προκύπτουν απο την διάδραση των τοπικών θεωριών περιβάλλοντος (context theories), χρησιμοποιώντας αναιρέσιμους κανόνες συσχέτισης και χρησιμοποιώντας μια σχέση προτίμησης *ανα πράκτορα* πάνω στα πολλαπλά περιβάλλοντα, η οποία αναπαριστάται ως ολική διάταξη επι των συστημάτων περιβάλλοντος.

Εδω, επεκτείνουμε αυτήν την προσέγγιση ώστε να χειρίζεται και περιπτώσεις όπου υπάρχει ελλειπής πληροφορία πάνω στις προτιμήσεις. Για να επιτευχθεί αυτό, αντικαθιστούμε στο μοντέλο της αναπαράστασης την ολική δίαταξη των προτιμήσεων με μία μερική διάταξη, και αναπροσαρμόζουμε το σημασιολογικό πλαίσιο και τους κατανεμημένους αλγόριθμους που είχαν προταθεί αρχικώς ώστε να ικανοποιούν τις νέες απαιτήσεις. Επιπλέον περιγράφουμε την αρχιτεκτονική μιας υλοποίησης σε μικρές συσκευές και παρουσιάζουμε την χρήση της με δύο σενάρια απο τους τομείς της Διάχυτης Νοημοσύνης και της Κοινωνικής κινητής Δικτύωσης. Τέλος, καταγράφουμε τις αρχικές μας εντυπώσεις από τον πειραματισμό μας με την Αναιρέσιμη Συλλογιστική Περιβάλλοντος (CDL) οπου έγινε χρήση κανονικών συσκευών σε πραγματικά περιβάλλοντα, και συζητούμε την απόδοση και τα θέματα επεκτασιμότητας που προκύπτουν από την προτεινόμενη προσέγγιση.

Επόπτης Καθηγητής:

Γρηγόρης Αντωνίου

Καθηγητής Τμήματος Επιστήμης Υπολογιστών

Πανεπιστημίου Κρήτης

# Ευχαριστίες

Αισθάνομαι την ανάγκη να ευχαριστήσω τον επόπτη καθηγητή μου, κύριο Γρηγόρη Αντωνίου για την πολύτιμη καθοδηγησή αλλα και για τις ουσιαστικές συμβουλές του με τις οποίες συνέβαλλε στην επιτυχή ολοκλήρωση των μεταπτυχιακών μου σπουδών. Θα ήθελα να ευχαριστήσω τον καθηγητή κύριο Δημήτρη Πλεξουσάκη για τις εύστοχες παρατηρήσεις και συμβουλές του ως μέλος της εισηγητικής επιτροπης της εργασίας οπως επίσης και τον επίκουρο καθηγητή κύριο Ιωάννη Τζίτζικα για τη συμμετοχή του στην εισηγητική επιτροπή. Εν συνεχεία, οφείλω να αναφερθώ στον διδάκτορα Αντώνη Μπικάκη που μου προσέφερε ανεκτίμητη βοήθεια καθ'ολη τη διάρκεια των μεταπτυχιακών σπουδών μου αλλα και για τη συμβολή του στην ολοκλήρωση της εργασίας μου. Οφείλω να ευχαριστήσω, τους φίλους μου και τους συμφοιτητές μου, για την ψυχολογική υποστήριξη, τη συντροφιά τους αλλα και τις αμέτρητες στιγμές χαράς που μου προσέφεραν κατα τη διάρκεια των σπουδών μου. Ολοκληρώνοντας, θέλω να ευχαριστήσω ιδιεταίρως τους γονείς μου Θοδωρή και Ιωάννα και την αδερφή μου Λουέλα για την συμπαράστασή τους, τη στήριξή που μου παρείχαν και εξακολουθούν να παρέχουν σε όλα μου τα βήματα. Σας ευχαριστώ.

**Στους γονείς μου Θοδωρή και Ιωάννα**

# List of figures

**Figure 1.1:** Context Information Flow in the Scenario ……..…....8
**Figure 2.1:** A magic box ...……..……...…..……..14
**Figure 3.1:** Arguments in the scenario.....32
**Figure 5.1:** CDL Layered Architecture ……....…....43
**Figure 5.2:** Technologies involved in a Query Operation…….…….....45
**Figure 5.3:** Sample user Profile published on web page…….…..…....46
**Figure 5.4:** Format of CDL query message ..…….……..…...48
**Figure 5.5:** Main Operations as Architectural Sub-Modules……..….....51
**Figure 5.6:** Query interaction using Bluetooth…….……..….53
**Figure 5.7:** Traces & Profiles of cell phone query interaction through sms …..…54
**Figure 6.1:** Execution tree of synthetic case of in breadth and in depth query…...70

# List of Tables

**Table 6.1:** Dr. Amber's profile…….……..……...57
**Table 6.2:** Common Part of User Profile….……..…58
**Table 6.3:** Knowledge Base loading time in msec…….…...62
**Table 6.4:** Local Query execution times…….…..64
**Table 6.5:** In breadth p2p query execution times…….…..…...66
**Table 6.6:** Estimated values in breadth p2p query performance……...…67
**Table 6.7:** In depth p2p query execution times…….69
**Table 6.8:** Synthetic case of p2p query execution……....70

# List of Charts

**Chart 6.1:** KB loading times mobile device vs emulator……..63

**Chart 6.2:** Local Query execution times, mobile device vs emulator……...65

**Chart 6.3:** In breadth p2p query performance, mobile device vs emulator……..66

**Chart 6.4:** Estimation of in breadth p2p query performance (simple query)……67

**Chart 6.5:** Estimation of in breadth p2p query performance (complex query).…68

**Chart 6.6:** In depth p2p query performance, emulator vs device……..69

**Chart 6.7:** Synthetic case of p2p query, emulator vs mobile device (estimate)....71

# Table of Contents

**List of Figures……xiii**

**List of Tables……...xv**

**List of Charts……xvii**

1. **Introduction..... 3**

1.1 Distributed Reasoning about Context in Ambient Intelligence .....3

1.2 Motivating Scenarios .....5

1.2.1 Social Mobile Computing Scenario .....5

1.2.2 Other Application Scenarios .....6

1.3 Approach .....9

1.4 Thesis Contribution .....10

1.5 Thesis Organization .....12

2. **Background ..... 13**

2.1 Contextual Reasoning in Artificial Intelligence .....13

2.1.1 Multi-Context Systems .....14

2.1.2 Non-monotonic Contextual Reasoning .....16

2.2 Argumentation Systems .....17

2.3 Preference-based Argumentation Systems .....19

2.4 Bikakis et al. approach [16] .....22

3. **Contextual Argumentation ..... 23**

3.1 Context Representation Model .....23

3.2 Argumentation Semantics .....25

3.2.1 Definitions .....26

3.2.2 Properties of the Framework .....30

4. **Distributed Algorithm for Query Evaluation ..... 33**
4.1 Algorithm Description ..... 33
4.2 Properties of the Algorithm ..... 40
4.3 Alternative Strategies ..... 41
5. **Contextual Defeasible Logic System ..... 43**
5.1 Architecture Overview ..... 43
5.2 Component Analysis and Implementation ..... 43
5.2.1 Application ..... 44
5.2.2 Profile and Knowledge Management ..... 45
5.2.3 Communications ..... 47
5.2.4 Reasoning and Inference ..... 49
5.3 Extensions and Optimizations ..... 51
6. **Experimental Evaluation ..... 55**
6.1 Technical Configuration of Scenarios ..... 55
6.2 Scenario based evaluation results ..... 60
6.3 General CDL System performance testing ..... 61
7. **Conclusions ..... 73**
7.1 Synopsis ..... 73
7.2 Future Directions ..... 74
8. **Appendix A ..... 77**
9. **Bibliography ..... 89**

# Chapter 1
# Introduction

Computing is moving towards pervasive, ubiquitous environments in which devices, software agents and services are all expected to seamlessly integrate and cooperate in support of human objectives, anticipating needs and delivering services in an anywhere, any-time and for-all fashion[78].

Ambient Intelligence systems aim to provide the right information to the right users, reached through the right device and of course at the right time and place. In order to achieve this, such a system must be fully aware of its environment, the people and devices that exist in it, their interests and capabilities and the ongoing or planned activities. All this information falls under the notions of context.

In computer science, context refers to the circumstances under which a device is being used. Context aware devices may also try to make assumptions about their current situation. The term context-awareness in ubiquitous computing was introduced by Schilit [66], where context is referred to as "location, identities of nearby people and objects, and changes to those objects".

Dey et al. [1] defines context as: “any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and application, including the user and applications themselves".

## 1.1 Distributed Reasoning about Context in Ambient Intelligence

The aim of reasoning about context in Ambient Intelligence and generally of a context-aware system is to deduce the meaning of low-lever raw context data e.g. from sensors, so as to perceive a situation as simple as an indication of temperature raise or as complex as recognizing an activity based also on other higher-lever available context information. Such a system, after gathering and processing all available information, raw or synthesized (based on defined rules), combines it and

ultimately translates it into valuable information. Based on this processed context information, the system can determine the state of its context and trigger actions in order to react appropriately to certain context changes. In order to achieve these, more and more sophisticated context models have been proposed, to support context-aware applications which use them to build smart environments.

In Ambient Intelligence, most systems so far, have followed fully centralized approaches (e.g. [29,34,36,47,48,53,62,76,77]), while others have used models based on the blackboard and shared memory paradigms (e.g. [52,54,55]). Collecting the reasoning tasks in a central entity certainly has many advantages. It achieves better control and better coordination between the participating entities. However, such solutions cannot meet the demanding requirements of ambient environments. The dynamics of the network and the unreliable and restricted (by the range of the transmitters) wireless communications call for fully distributed solutions.

Considering the above we propose reasoning with multiple contexts. This is a combination of local reasoning and distributed reasoning. Local reasoning is performed using knowledge of a single context. Local conclusions in a given context are derived from a set of axioms and inference rules that model local context knowledge, and which constitute a small subset of the entire knowledge, while distributed reasoning takes into account the possible relations between local contexts. These relations result from the fact that different contexts actually constitute different representations of the same world. They are modelled as inference rules (known as mapping or bridge rules) with premises and consequences in different contexts, and enable information flow between related contexts.

Context information, however, can be unknown, imprecise, erroneous, or even ambiguous. These four types Henricksen and Indulska characterize as context imperfections in [49]. Connectivity failures in wireless networks sometimes result in *unknown-unavailable* context data. Imprecision is an affliction of sensor-derived information, while erroneous context arises usually as a result of human or hardware errors. Lastly, when data about a context property comes from multiple sources, then context may become ambiguous. In the same time, entities operating in an Ambient Intelligence environment are expected to have different goals, experiences and perceptive capabilities.

Overall, the role of reasoning about context in Ambient Intelligent systems includes:

- Transforming the low level raw context data into higher level meaningful information so that it can be later used in the application layer
- Making decisions about the system behavior when certain changes are detected in the system's context.

In order for the system to do so, it has to evaluate the quality and the validity of the sensed data, detect possible errors in the context information and handle missing values, generally to be able to do distributed reasoning with highly dynamic and imperfect context in a real-time fashion, taking into consideration the restricted computational capabilities of some mobile devices and the constraints of wireless communications. Overall, the intelligence of a system is mainly determined by its reasoning capabilities.

## 1.2 Motivating Scenarios

In this section we present the main motivating scenario, this scenario is used as an example throughout chapter 3 and chapter 4 for demonstrating the theoretic points of our approach. In addition we describe two application scenarios, highlighting different aspects. The first one focuses more on social interactions, while the second highlights more on the integration of information from various devices (contexts). In chapter 6 there is a full technical description of these two interesting types of scenarios that were used for the evaluation of the proposed methods, but also served as motivations for our research.

### 1.2.1 Social Mobile Computing Scenario

This scenario involves a student, Adam, using a cell phone and takes place at University of Crete, Greece. An SMS is received by Adam's cell phone at 10 am from

a multiplex cinema to inform him that a new action movie is available at 10pm that night at the open air cinema by the beach. Based on Adam's profile information that it carries and on knowledge about Adam's context, the mobile phone must determine whether Adam should be informed about the movie.

Adam' profile contains rules dictating activity notification based on information involving his schedule, weather, personal interests and university obligations.

Adam's preferences about movies are included in his profile. Adam has also specified two web sites – meteo.gr and poseidon.gr – as external sources for information about weather conditions. Meteo.gr describes general weather conditions, while poseidon.gr specializes more on wind reports. For class related information, the mobile phone contacts the University of Crete web server that retains the schedule of all classes. Finally, the external sources used for information about movies are IMDB and Rotten Tomatoes. Adam has also specified that he considers poseidon.gr more trustworthy than meteo.gr, while when it comes to movie-related information, he prefers IMDB to Rotten Tomatoes.

For the specific scenario, we assume that Adam is interested in action movies. Knowledge imported from meteo.gr indicates good weather conditions for an outdoor activity, while poseidon.gr claims the opposite regarding winds. Based on class information imported from the University of Crete web server, there is no conflict for the date and time in question. Finally, we assume that IMDB suggests it's a good movie, while Rotten Tomatoes holds the opposite view.

### 1.2.2 Other Application Scenarios

Our aim was to develop a distributed framework that may support various types of context-aware applications. The first one demonstrates the integration of knowledge by a Mobile phone, using our proposed methods, from various contexts (e.g. sensors) and reasons with it in order to figure out something for itself, a *pull* type of scenario. The other one involves a mobile device that is being queried by other known or unknown sources. It is being *pushed* information in the form of a query

(e.g. for "social events") so as to decide if it is interested. Therefore it shows further applications in social computing, advertising etc.

### 1.2.2.1 Social Networking Scenario

The concrete scenario takes place in Heraklion and involves three students carrying their personal mobile phones. Consider that user A is at the University, user B is downtown, while user C is at FORTH (all three in different parts of the city).

All three users have their profiles loaded on their mobile phones, share a common interest in the domain of Semantic Web, and are enrolled in the CS585 class. In addition, B has a public profile online on the university server, A and B have a common interest in tennis and are university friends, as are B and C. Finally, all three have the same preferences for being notified: generally they want to receive information when it is within their areas of interest. However, they do not want to be notified of leisure events when they are scheduled for work at the same time.

Assume that A passes by the University Front Desk, and its Bluetooth server advertises that the lesson of CS585 scheduled to take place later on that day is canceled. A is notified, and his mobile phone automatically forwards this announcement to A's university friends via SMS. Consequently, B is also notified, and in turn his mobile forwards the information to his university friends. C, who is a university acquaintance of B, is notified as well.

At the same time, C passes by the ICS-FORTH Lobby and gets informed by its Bluetooth server of a lecture about Semantic Web, and a tennis tournament both taking place at FORTH. Again, all users are informed of the lecture, but only B is informed of the tennis tournament: C has no interest in tennis, while A has to work on the same day with the event.

Finally, users check periodically, and when Internet access is available, their friends' online profiles for birthdays, and discover that the current day it is B's birthday; therefore A and C are notified about it.

### 1.2.2.2 Context-Aware mobile phone in Ambient Classroom Scenario

The scenario involves a context-aware mobile phone that has been configured by Dr. Amber to decide whether it should ring (in case of incoming SMS) based on his preferences and context. Dr. Amber has the following preferences: His phone should ring in case of an incoming SMS, unless it is in silent mode or he is giving a lecture.


**Figure 1.1:** Context Information Flow in the Scenario

Suppose Dr. Amber is currently located in the ’RA201’ classroom. It is class time, but he has just finished with a course lecture, and still remains in the classroom reading his emails on his laptop. The mobile phone receives an incoming SMS, while it is in normal mode.

To decide whether it should ring, the phone requires a number of context parameters related to Dr. Amber’s current activity. Therefore, it attempts to contact through the wireless network of the university other ambient agents that are located nearby, import from them further context information, and use this information to reach a decision.

To determine whether Dr. Amber is currently giving a lecture, the mobile phone uses two rules. The first rule states that if at the time of the call there is a scheduled lecture, and Dr. Amber is located in a university classroom, then he is possibly giving a lecture. Information about scheduled events is imported from Dr. Amber's laptop, which hosts his calendar. According to this, there is a scheduled class event at the time of the sms arrival. Information about Dr. Amber's current position is imported from the wireless network localization service, which at the time of the SMS arrival localizes Dr. Amber (actually his mobile phone) in 'RA201' university classroom. The second rule states that if there is no class activity in the classroom, then Dr. Amber is rather not giving a lecture.

Information about the state of the classroom is imported from the classroom manager, a stationary computer installed in 'RA201'. In this case, based on information about the presence of people in the classroom that imports from an external person detection service, it detects only one person, therefore the classroom manager infers that there is no class activity.

The overall information flow in the scenario is depicted in Figure 6.1. Eventually, the mobile phone will receive ambiguous information from the various ambient agents operating in the classroom. Information imported from Dr. Amber's laptop and the localization service leads to the conclusion that Dr. Amber is currently giving a lecture, while information imported from the classroom manager leads to the contradictory conclusion that Dr. Amber is not currently giving a lecture. To resolve this conflict, the mobile phone must be able to evaluate the information it receives from the various sources. For example, in case it is aware that the information derived from the classroom manager is more accurate than the information imported from Dr. Amber's laptop, it will determine that Dr. Amber is not currently giving a lecture, and therefore reach the 'ring' decision.

## 1.3 Approach

Our work is based on previous work by Bikakis et al. [1], which proposed a framework using as underlying language that of Defeasible Logic. It takes into account the distribution of context information using the notion of mapping rules in addition to the local ones, and a per-context total order preference relation on the

system contexts. It uses arguments, which are constructs based on combinations of rules, to support a decision. Decisions derived from information from more preferred contexts are considered to be stronger than those that use information from less preferred contexts. So in case of conflict a context is able to choose accordingly what to believe. Finally it handles local inconsistencies using defeasible logic and uses the context preference relation for conflict resolution occurring in global scale through the use of mapping rules.

## 1.4 Thesis Contribution

This thesis presents the implementation on mobile devices of a fully distributed approach for reasoning about context in Ambient Intelligence environments. It is called Contextual Defeasible Logic (CDL). It is based on completed conceptual works adopting ideas from Defeasible Logic [7] and Multi-Context Systems (MCS). MCS were initially introduced in [42], where they were defined as logical formalizations of distributed context theories connected through a set of inference rules (known as mapping or bridge rules) that enable information flow between different contexts. In CDL, the original model was extended with local defeasible theories, defeasible bridge rules and a *per-context* preference relation, representing the confidence of a context in its information sources. These features have enabled handling cases of uncertain local knowledge, but also resolving possible ambiguities that may arise from the interaction of contexts through their mappings. The basic model and associated algorithms were presented in [19], an argumentation-theoretic semantics was introduced in [20], and its relevance to the area of Ambient Intelligence was analyzed in [18].

In the original approach, perfect preference information was assumed in the sense that (a) all agents need to be compared to each other, and (b) that each agent has a total preference relation on all fellow agents. However, this requirement may not by realistic in uncertain environments. In Ambient Intelligence environments an agent may collect information of various types from several different information sources, which cannot be compared in terms of confidence. It is not, for example, rational to compare a localization service with a weather forecast service. To handle this problem, we extend the original theoretical framework as follows:

- We extend the MCS model of Bikakis et al. to additionally deal with incomplete preference information. To enable this, the total preference ordering is replaced with partial ordering, and the argumentation framework and the distributed algorithms are accordingly modified to meet the new requirements.

- We provide a new strategy for distributed query evaluation, which propagates the preferences of the context that initiates the query. This way the original preferences are used throughout the distributed query evaluation process, minimizing the number of messages circulating, and enhancing the distribution of the reasoning tasks.

Other major contributions of this work are results of its practical part, which mainly involves:

(a) The development of an infrastructure for *small devices*, that: (i) integrates an existing prolog engine [31] and an api for facilitating the access to the reasoning tasks, (ii) provides protocols for various communication technologies and (iii) among others, utility functions for handling profile and knowledge and user interfaces, all "tailored" to support both local defeasible reasoning, using a lightweight version of the metaprogram, and distributed reasoning through the implementation of distributed query evaluation algorithms from [20].

(b) The implementation of real application scenarios, aiming
   - To illustrate the MCS context model, the argumentation semantics and to analyze the distributed query evaluation algorithms.
   - To demonstrate other uses of the implemented infrastructure based on "pushing" information.

(c) An evaluation of the infrastructure based not only on simulations but also on real devices in real AmI environments. For this purpose two of the scenarios were fully defined, implemented and finally deployed on actual

devices, which provided interesting results that are analyzed and discussed further.

## 1.5 Thesis Organization

The rest of the Thesis is organized as follows:

Chapter 2 provides background information on Contextual Reasoning and Argumentation Systems. It presents the concept of Multi Context Systems and discusses approaches for integrating preferences in Argumentation Systems.

Chapter 3 describes the argumentation framework that enables reasoning with the imperfect and distributed context information. Firstly, it presents the MCS-based representation model, and then provides its semantic characterization using arguments.

Chapter 4 provides the operational model in the form of a distributed algorithm for query evaluation, and presents its formal properties regarding its termination, complexity, soundness and completeness with respect to the argumentation framework. It also briefly describes other three alternative strategies for conflict resolution.

Chapter 5 presents the architecture of the CDL System and provides the role of each component and the details of their interaction.

Chapter 6 describes the implementation of two application scenarios as well as the setup of the experiments that we conducted, and provides the results of the performance evaluation of the CDL System.

Chapter 7 summarizes the main points of the thesis and discusses potential extensions of this work.

# Chapter 2
# Background

This chapter provides background information on Contextual Reasoning in Artificial Intelligence, introduces the notion of Argumentations Systems and describes the intuition behind Multi-Context Systems. Finally it presents non monotonic approaches and discusses ideas on the integration of preferences in Argumentation Systems.

## 2.1 Contextual Reasoning in Artificial Intelligence

McCarthy introduced in [56] the notion of context and contextual reasoning in Artificial Intelligence. He argued there that the combination of non-monotonic reasoning and contextual reasoning would constitute an adequate approach for the solution of the problem of generality. Two main formalizations have been proposed since then to formalize context: the Propositional Logic of Context (PLC [27, 57]), and the Multi-Context Systems introduced in [42], which later became associated with the Local Model Semantics proposed in [39]. Multi-Context Systems has been argued to be most adequate with respect to the three properties of contextual reasoning (partiality, approximation, proximity ) as these were formulated by [15] and has been shown to be technically more general than PLC [67].

Context reasoning mechanisms have already been used in practice in Artificial Intelligence and are expected to play a significant role in the development of the next generation Applications. The Semantic Web is the most representative example of distributed knowledge [10,75]. It provides well-defined meaning to information and therefore it can be viewed as partial theories spread all over the Internet that applications can collect and reason on. Multi-Context Systems have been also used as the underlying model for agent architectures, as for example in [59] and [65]. Both studies propose breaking the logical description of an agent into a set of contexts, each of which represents a different component of the architecture, and the interactions between these components are specified by means of bridge rules

between the contexts. Other fields that the notions of context and contextual reasoning have been successfully applied include: multi-agent systems [13, 30], modeling dialog, argumentation and information integration in electronic commerce applications [59], commonsense reasoning [24], reasoning about beliefs [14, 33, 38, 40, 41] and reasoning with viewpoints [11].

### 2.1.1 Multi-Context Systems

The underlying idea behind Multi Context Systems is highlighted by the following example called the 'magic box':

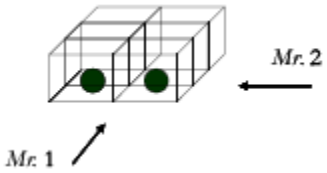



**Figure 2.1:** A magic box.

As shown in Figure 2.1, Mr.1 and Mr.2 both observe the same box; it is called ‘magic’ because neither of them can make out its depth. Both Mr.1 and Mr.2 have their beliefs about the box, perhaps in different representation. Mr.1's beliefs may have to do with concepts that are completely meaningless for Mr.2, and vice versa. For instance, Mr.2 could believe the central section of the box to contain a ball. From Mr.1's point of view however, the box does not have a central section, so any statement about whether it contains a ball or not is meaningless for him. Mr.1 and Mr.2 may also share some concepts, but in any case their respective interpretations of those concepts are independent. For instance, "the right section of the box contains a ball" is a meaningful statement for both Mr.1 and Mr.2. But it is perfectly conceivable that Mr.1 believes the right section of the box to be empty, while Mr.2 believes it to contain a ball, and vice versa.

The bottom line is that both observers have their own local language in which they express their beliefs and the challenge is to bridge that gap. More specifically, if observers were to have access to each other's beliefs and know about the language used, Mr.1 for instance, could access the fact that Mr.2 believes that the box contains

a ball and use this knowledge to map this fact through rules in his own language and adapt his beliefs accordingly. We think of this procedure as *information flow* among different observers.

The most important characteristics of Multi-Context Systems can be summarized in the following points.

- A context is a subset of an individual global state, or - more formally - a partial and approximate theory of the world from some individual's perspective [40]. As a consequence, contexts are expected to use different languages, data representation and inference systems (heterogeneity issue).

- Reasoning with multiple contexts is a combination of:

    - Local reasoning, which is performed using knowledge of a single context. Local conclusions in a given context are derived from a set of axioms and inference rules that model local context knowledge, and which constitute a small subset of the entire knowledge.

    - Distributed reasoning, which takes into account the possible relations between local contexts. These relations result from the fact that different contexts actually constitute different representations of the same world. They are modeled as inference rules (known as mapping or bridge rules) with premises and consequences in different contexts, and enable information flow between related contexts.

- Although each context may be locally consistent, global consistency cannot be assumed, required or guaranteed. As a consequence, potential conflicts may arise from the interaction of contexts through their mappings.

- The relationship between different contexts can be described only to a partial extent. In other words, no context can be 'fully' translated into another, as each of them may encode assumptions that are not fully explicit.

### 2.1.2 Non-monotonic Contextual Reasoning

Multi-Context Systems have been the basis of two recent studies that were the first to deploy non-monotonic features in contextual reasoning: (a) The non-monotonic rule-based MCS framework [63] and (b) the multi-context variant of Default Logic, ConDL [26].

These approaches support all fundamental characteristics for contextual reasoning in Ambient Intelligence environments: *partiality*, *approximation* and *perspective*, which are inherently supported by the generic MCS model. Both also support some additional desired characteristics: support for reasoning with incomplete information using default negation in the body of the mapping rules.

Furthermore, Contextual Default Logic handles ambiguous context using default mapping rules. The case that context A imports conflicting context information from contexts B and C through A's mapping rules, is modeled using the different extensions of the theory that includes A's local theory, A's mapping (default) rules, the local theories and (possibly) the mappings of contexts B and C, and (possibly) other context theories that B and C are connected to through their mapping rules.

Finally, compared to [63], the ConDL approach has the additional advantage that is closer to implementation due to the well-studied relation between Default Logic and Logic Programming.

However, there are still some issues that Contextual Default Logic cannot efficiently handle. More specifically, it does not provide ways to model the quality of the imported context information, nor the preference between two different information sources. In other words, it does not include any notion of priority that would allow resolution of potential conflicts that may arise while importing context information from two different sources. Furthermore, computing extensions or checking if a formula is in one or in all extensions of a Default theory has been showed to be a complex computational problem [44, 70], and would add a too heavy computational overhead to the devices operating in Ambient Intelligence environments, which typically have limited computational capabilities.

## 2.2 Argumentation Systems

Argumentation systems constitute a way to formalize non-monotonic reasoning, viz. as the construction and comparison of arguments for and against certain conclusions. In these systems, the basic notion is not that of a defeasible conditional but that of a defeasible argument. The idea is that the construction of arguments is monotonic, i.e., an argument stays an argument if more premises are added. Non-monotonicity, or defeasibility, is not explained in terms of the interpretation of a defeasible conditional, but in terms of the interactions between conflicting arguments: in argumentation systems non-monotonicity arises from the fact that new premises may give rise to stronger counter-arguments, which defeat the original argument.

Argumentation systems can be applied to any form of reasoning with contradictory information, whether the contradictions have to do with rules and exceptions or not. For instance, the contradictions may arise from reasoning with several sources of information, or they may be caused by disagreement about beliefs or about moral, ethical or political claims. Moreover, it is important that several argumentation systems allow the construction and attack of arguments that are traditionally called ampliative, such as inductive, analogical and abductive arguments; these reasoning forms fall outside the scope of most other non-monotonic logics.

Most argumentation systems have been developed in artificial intelligence research on non-monotonic reasoning, although Pollock's work [60], which was the first logical formalization of defeasible argumentation, was initially applied to the philosophy of knowledge and justification (epistemology). The first artificial intelligence paper on argumentation systems was [64]. Argumentation systems have been applied to domains such as legal reasoning, medical reasoning and negotiation. Below, we present an abstract approach to defeasible argumentation, developed in several articles by Bondarenko, Dung, Toni and Kowalski. We also give a brief description of some other interesting approaches.

### The Abstract Approach of Bondarenko, Dung, Kowalski and Toni

Historically, this work came after the development by others of a number of argumentation systems (to be discussed below). The major innovation of the BDKT

approach is that it provides a framework and vocabulary for investigating the general features of these other systems, and also of non-monotonic logics that are not argument-based.

The latest and most comprehensive account of the BDKT approach is Bondarenko et al. [23]. In this account, the basic notion is that of a set of assumptions. In their approach the premises come in two kinds: ordinary premises, comprising a theory, and assumptions, which are formulas (of whatever form) that are designated (on whatever ground) as having default status. Bondarenko et al. regard non-monotonic reasoning as adding sets of assumptions to theories formulated in an underlying monotonic logic, provided that the contrary of the assumptions cannot be shown. What in their view makes the theory argumentation-theoretic is that this provision is formalized in terms of sets of assumptions attacking each other. In other words, according to [23], an argument is a set of assumptions. This approach has especially proven successful in capturing existing non-monotonic logics.

Another version of the BDKT approach, presented by Dung [32], completely abstracts from both the internal structure of an argument and the origin of the set of arguments; all that is assumed is the existence of a set of arguments, ordered by a binary relation of defeat. Dung then defines various notions of so-called argument extensions, which are intended to capture various types of defeasible consequence. These notions are declarative, just declaring sets of arguments as having a certain status. Finally, Dung shows that many existing non-monotonic logics can be reformulated as instances of the abstract framework.

**Governatori et al.**

In [46], Governatori et al. provide argumentation semantics for two defeasible logics: ambiguity blocking and ambiguity propagating Defeasible Logic [9]. In both cases they disregard the rule superiority relation without, however, affecting the generality of their approach, as it has been proved that this relation can be simulated by the other ingredients of the logic [7]. Specifically, they show that Dung's grounded semantics characterizes the ambiguity propagating Defeasible Logic, while to give a semantic characterization of the original (ambiguity blocking) Defeasible Logic they modify

Dung's notion of acceptability. In this framework, they define arguments as (possible infinite) proof trees, and classify them into two categories: strict and defeasible arguments. They define the notions of attack and undercut for defeasible arguments only. Specifically, a defeasible argument is attacked by arguments with conflicting conclusion, and undercut when a proper subargument is attacked by a justified argument. They also define the notions of acceptable, justified and rejected arguments for the two logics, taking into account the attacks against them. The argumentation framework for ambiguity blocking Defeasible Logic of [46] can be considered as a special case of the argumentation theoretic semantics of the ambiguity blocking DL with superiority relation provided in [45].

A distinct characteristic of the latter approach is that in order to handle team defeat, which refers to the full Defeasible Logic with priorities [9], they define arguments as sets of proof trees, instead of single proof trees.

## 2.3 Preference-based Argumentation Systems

A central notion in argument-based reasoning is that of acceptability of arguments. In general, to determine whether an argument is acceptable, it must be compared to its counter-arguments; namely, those arguments that support opposite or conflicting conclusions. In preference-based argumentation systems, this comparison is enabled by a preference relation, which is either implicitly derived from elements of the underlying theory, or is explicitly defined on the set of arguments. In the works of Simari and Loui [68], and Stolzenburg et al. [71], the preference relation takes into account the internal structure of arguments. Overall, arguments are compared in terms of specificity. Intuitively, the notion of specificity favors two aspects in an argument; it prefers an argument with greater information content or less use of defeasible information. In other words, an argument will be deemed better than another if it is more precise or concise. The object language used in [68] is a first-order language, which uses facts and defeasible rules, while in the case of [71], the underlying language is Defeasible Logic Programming [37], which uses both strict and defeasible rules. An extension of [68], which explicitly supports distributed entities in the argumentation process, and an argumentation-based Multi-Agent Systems

architecture have been recently proposed by Thimm et al. in [72, 74]. In this framework, all agents share common strict knowledge, which is encoded in strict rules, while each of them also possesses subjective beliefs encoded in local defeasible theories. Each agent generates its own arguments on the basis of the global knowledge and the local belief bases and reacts on arguments of other agents with counterarguments. In [73], they extend this framework to support collaborations between agents, which enable combining the belief bases of different agents and producing more and better arguments. The distributed nature of these frameworks is, however, compromised by two assumptions that they make; namely, the existence of a meta-agent representing and acting on behalf of collaborative agents, and the disjointness of collaborations - one agent cannot participate in more than one collaborations.

The approaches followed by Prakken and Sartor [61], Governatori et al. [45], and Kakas and Moraitis [51] share the common feature that preferences among arguments are derived from a priority relation on the rules in the underlying theory. Specifically, in [61], the underlying language is that of Extended Logic Programming, which includes strict rules and defeasible rules, while preference information is provided in the form of an ordering on the defeasible rules. They study two different cases with respect to the nature of this ordering. In the first one, the ordering is fixed and indisputable (strict priorities), while in the second case priorities are themselves defeasibly derived as conclusions within the system. To support defeasible priorities, they allow stating rules and constructing arguments about priorities. The argumentation framework of Governatori and Maher [45] uses the language of Defeasible Logic. In this framework, the rule priority relation of Defeasible Logic is used to determine whether an argument is defeated by a counter-argument. Similarly with [61], the framework proposed by Kakas and Moraitis [51] also includes the notion of dynamic priorities. Specifically, the underlying monotonic logic includes a special type of rules that are used to give priority between competing rules in case of conflict. Based on these rules, they build arguments on priorities, and reason with them to give preference to specific arguments in the system. Another interesting feature of this work is the integration of abduction to handle cases of incomplete information.

An alternative approach, which is followed by Bench-Capon [12], and by Kaci and Torre [50], relates arguments to the values they promote. In the so called Value Based Argumentation Frameworks, the preference ordering of the arguments is derived from a preference ordering over their related values. In [12], each argument promotes only a single value, and an argument is preferred to another one if and only if the value promoted by the former argument is preferred to the value promoted by the latter one.

Kaci and Torre [50] provide a generalization of this approach, in which arguments can promote multiple values and the comparison is conducted between sets of arguments. The abstract preference-based argumentation frameworks proposed by Amgoud and her colleagues [3, 4, 5] assume that preferences among arguments are induced by a preference relation defined on the supports of the arguments, which is a partial preordering. The preference relation on the supports is itself induced by a priority relation defined on the underlying belief base. In [6], Amgoud et al. introduce the notion of contextual preferences in the form of several pre-orderings on the belief base, to take into account preferences that depend upon a particular context. Conflicts between preferences may appear when these preferences are expressed in different contexts; e.g. an argument A may be preferred to another argument B in a context $C_1$ and the argument B may be preferred to A in a context $C_2$. To resolve this kind of conflicts, they use meta-preferences, in the form of total preordering on the set of contexts.

The abstract argumentation framework of Modgil [58] integrates meta-level argumentation about preferences in Dung's argumentation theory. The extended framework retains the basic conceptual machinery of Dung's theory, and extends the notion of defeat to account for arguments that claim preferences between arguments.

Finally, building on previous works on inconsistency handling in P2P Query Answering Systems [2, 22, 28], Binas and McIlraith propose an argumentation framework that integrates preference information about the system peers [21]. They assume that each peer has its own knowledge base encoded in propositional logic, and use a preference ordering on the system peers to resolve potential conflicts that may arise between arguments derived from different peers.

## 2.4 Bikakis et al. approach [16]

This argumentation framework is an extension of the framework of Governatori et al. [46]. Both frameworks use the language of Defeasible Logic without superiority relation as their underlying language. To take into account the distribution of context information, and a preference ordering on the system contexts, this framework additionally uses the notions of local and mapping arguments, argumentation lines, and ranks of arguments. In the proposed framework, preferences are derived both from the structure of arguments - arguments that use strict rules are considered stronger than those that use defeasible rules - and from the preference ordering - arguments that use information from more preferred contexts are stronger than those that use information from less preferred contexts. Among the remaining works that we discussed in this section, this approach shares common ideas with the argumentation framework with contextual preferences of Amgoud et al. [7], in the sense that both frameworks allow different contexts to define different preference orderings. The main differences are that in this case, these orderings are applied to the contexts themselves rather than directly to a set of arguments, and that each context uses a different knowledge base; and therefore a different set of arguments. Compared to the P2P Argumentation System of Binas and McIlraith [21], [16] framework differs in two primary ways: (a) it can additionally handle local inconsistencies using defeasible arguments; (b) it assumes that each context uses its own total ordering preference, opposed to the global preference relation used in [21].

# Chapter 3

# Contextual Argumentation

In this section we present an argumentation framework for Contextual Defeasible Logic using Bikakis et al. [17] approach which we extended by relaxing the requirements of total order on preferences, thus we were able to support partial trust relationships. More specifically, using the motivating scenario from the introduction we explain the changes to the representation model and the argumentation semantics of the proposed framework.

## 3.1 Context Representation Model

We model a Multi-Context System $C$ as a collection of distributed context theories $C_i$: A context is defined as a tuple of the form$(V_i, R_i, T_i)$ where $V_i$ is the vocabulary used by $C_i$, $R_i$ is a set of rules, and $T_i$ is a preference relation on $C$.

$V_i$ is a set of positive and negative literals. If $q_i$ is a literal in $V_i$, $\sim q_i$ denotes the complementary literal, which is also in $V_i$. If $q_i$ is a positive literal p then $\sim q_i$ is $\neg p$; and if $q_i$ is $\neg p$, then $\sim q_i$ is $p$. We assume that each context uses a distinct vocabulary.

$R_i$ consists of two sets of rules: the set of local rules and the set of mapping rules. The body of local rules is a conjunction of local literals (literals that are contained $V_i$), while their head contains a single local literal. There are two types of local rules:

- Strict rules, of the form: $r_i^l: a_i^1, a_i^2, \dots a_i^{n-1} \rightarrow a_i^n$ . They express local knowledge and are interpreted in the classical sense: whenever the literals in the body of the rule are strict consequences of the local theory, then so is the literal in the head of the rule. Local rules with empty body denote factual knowledge.
- Defeasible rules, of the form: $r_i^d: b_i^1, b_i^2, \dots b_i^{n-1} \Longrightarrow b_i^n$. They are used to express uncertainty, in the sense that a defeasible rule cannot be applied to support its conclusion if there is adequate contrary evidence.

Mapping rules associate local literals with literals from the vocabularies of other contexts (foreign literals). The body of each such rule is a conjunction of local and foreign literals, while its head contains a single local literal: $r_i^m$: $a_i^1, a_j^2, \dots a_k^{n-1} \Longrightarrow a_i^n$. $r_i^m$ associates local literals of $C_i$ (e.g. $a_i^1$) with local literals of $C_j$ ($a_j^2$), $C_i$( $a_k^{n-1}$) and possibly other contexts. $a_i^n$ is a local literal of the theory that has defined $r_i^m$ ($C_i$).

Finally, each context $C_i$ defines a preference relation which is a partial order. In the following chapters we use a certain representation form which is also used in the algorithm in chapter 4. It is a partial order $T_i$ on $C$ and expresses its confidence in the knowledge it imports from other contexts. Represented as a set: $T_i = \{ L_j\}$, where $L_j$ is a list representing a path of the partial order tree in which when a context $C_k$ precedes another context $C_l$ then $C_k$ is preferred over $C_l$.

**Example.** The scenario described in the introduction (section 1.2.1) can be represented by the following context theories. Rules $r_{11}$ to $r_{110}$ constitute the context theory of the mobile phone (represented as context $C_1$), rules $r_{21}^l$ and $r_{22}^l$ constitute the context theory of meteo.gr ($C_2$), while rules $r_{31}^l$, $r_{41}^l$, $r_{51}^l$ and $r_{61}^l$ constitute respectively the context theories of poseidon.gr ($C_3$), the university web server ($C_4$), imdb.com ($C_5$) and Rotten Tomatoes ($C_6$). Note that through rules $r_{16}$ to $r_{110}$, the mobile phone imports context information from the available external sources. $T_1$ represents the preference ordering defined by $C_1$.

- $r_{11}^d$: $\text{goodWeather}_1, \text{freeSchedule}_1, \text{likesMovie}_1, \text{goodMovie}_1 \Longrightarrow \text{activity}_1$
- $r_{12}^d$: $\text{badWeather}_1 \Longrightarrow \neg\text{activity}_1$
- $r_{13}^d$: $\text{badMovie}_1 \Longrightarrow \neg\text{activity}_1$
- $r_{14}^l$: $\text{actionMovie}_1 \rightarrow \text{likesMovie}_1$
- $r_{15}^l$: $\rightarrow \text{actionMovie}_1$
- $r_{16}^m$: $sunny_2, high\_temp_2 \Longrightarrow \text{goodWeather}_1$
- $\text{r}_{17}^{\text{m}}$: $\text{windy}_3 \Longrightarrow \text{badWeather}_1$
- $\text{r}_{18}^{\text{m}}$: $\neg\text{scheduledlesson}_4 \Longrightarrow \text{freeSchedule}_1$
- $\text{r}_{19}^{\text{m}}$: $\text{highratedMovie}_5 \Longrightarrow \text{goodMovie}_1$

- $\mathrm{r}^{\mathrm{m}}_{110}$: $\mathrm{lowratedMovie}_6 \Longrightarrow \mathrm{badMovie}_1$

- $T_1 = \{[C_3, C_2], [C_5, C_6]\}$

- $r^l_{21}: \rightarrow sunny_2$
- $r^l_{22}: \rightarrow high\_temp_2$
- $r^l_{31}: \rightarrow windy_3$
- $r^l_{41}: \rightarrow \neg scheduledlesson_4$
- $r^l_{51}: \rightarrow highratedMovie_5$
- $r^l_{61}: \rightarrow lowratedMovie_6$

## 3.2 Argumentation Semantics

The argumentation framework described in this section extends the argumentation semantics of Defeasible Logic [46] with the notions of distribution of the available information, and preference among system contexts. It modifies the argumentation framework in [17] by relaxing the requirements of total preference order.

The framework uses arguments of local range, in the sense that each one contains rules of a single context only. Arguments of different contexts are interrelated in the Support Relation (Definition 1) through mapping rules. The relation contains triples; each triple contains a proof tree for a literal of a context using the rules of the context. The proof tree may contain either only local rules, or both local and mapping rules. In the second case, for a triple to be contained in the Support Relation, similar triples for the foreign literals of the triple must have already been obtained. We should also note that for sake of simplicity we assume that there are no loops in the local context theories, and thus proof trees are finite. However the global knowledge base may contain loops caused by mapping rules, which associate different context theories.

### 3.2.1 Definitions

**Definition 1.** Let $C = \{C_i\}$ be a Defeasible MCS. The Support Relation of C ($SR_C$) is the set of all triples of the form $(C_i, PT_{p_i}, p_i)$, where $C_i \in C, p_i \in V_i$, and $PT_{p_i}$, is the proof tree for $p_i$ based on the set of local and mapping rules of $C_i$. $PT_{p_i}$, is a tree with nodes labeled by literals such that the root is labeled by $p_i$, and for every node with label q:

1. $If\ q \in V_i\ and\ a_1, \dots, a_n\ label\ the\ children\ of\ q\ then$

   $-\ If\ \forall a_i (i = 1, \dots, n): a_i \in V_i\ then\ there\ is\ a\ local\ rule\ r_i \in C_i\ with\ body\ a_1, \dots, a_n and\ head$

   $-\ If\ \exists a_j\ such\ \ that\ a_j \notin V_i\ then\ there\ is\ a\ mapping\ rule\ r_i \in C_i\ with\ body\ a_1, \dots, a_n and\ head\ q$

2. $If\ q\ \in V_j \neq V_i, then\ this\ is\ a\ leaf\ node\ of\ the\ tree\ and\ there\ is\ a\ triple\ of\ the\ form\ (C_j, PT_q, q\ )\ in\ SR_C$
3. $The\ arcs\ in\ a\ proof\ tree\ are\ labeled\ by\ the\ rules\ used\ to\ botain\ them.$

**Definition 2.** $An\ argument\ A\ for\ a\ literal\ p_i is\ a\ triple\ (C_i, PT_{p_i}, p_i)\ in\ SR_C.$

Any literal labelling a node of $PT_{p_i}$ is called a *conclusion* of A. However, when we refer to the *conclusion* of A, we refer to the literal labelling the root of $PT_{p_i}$ ($p_i$). We write $r \in A$ to denote that rule r is used in the proof tree of A.

The definition of *subarguments* is based on the notion of subtrees.

**Definition 3.** *A (proper) subargument of A is every argument with a proof tree that is (proper) subtree of the proof tree of A.*

Based on the literals contained in their proof tree, arguments are classified to local arguments and *mapping* arguments. Based on the type of the rules that they use, local arguments are either *strict* local arguments or *defeasible* local arguments.

**Definition 4.** *A local argument of context $C_i$ is an argument that contains only local literals of $C_i$. If a local argument A contains only strict rules, then A is a strict local argument; otherwise it is a defeasible local argument. A is mapping argument if its proof tree contains at least one foreign literal.*

**Definition 5.** *$Args_{C_i}$ is the set of arguments derived from context $C_i$. $Args_C$ is the set of all arguments in $C$: $Args_C = \cup Args_{C_i}$.*

The conclusions of all strict local arguments in $\text{Args}_{\text{C}_\text{i}}$ are logical consequences of $\text{C}_\text{i}$. Distributed logical consequences are derived from a combination of local and mapping arguments in $\text{Args}_\text{C}$. In this case, we should also consider conflicts between competing rules, which are modelled as attacks between arguments, and preference orderings, which are used in our framework to compare mapping arguments.

**Definition 6.** *An argument A attacks a defeasible local or mapping argument B at $p_i$, if $p_i$ is a conclusion of B, $\sim p_i$ is a conclusion of A, and the subargument of B with conclusion $p_i$ is not a strict local argument.*

**Definition 7.** *An argument A defeats a defeasible or mapping argument B at $p_i$, if A attacks B at $p_i$, and for the subarguments of A, $A'$ with conclusion $\sim p_i$, and of B, $B'$ with conclusion $p_i$, it holds that:*

1. $\text{B}'$ is not a strict local argument and either
2. (a) $\text{A}'$ is a local argument of $\text{C}_\text{i}$ or
   (b) $B' is\ a\ mapping\ argument\ of\ C_i$ and

$$\exists b_l \in B : \forall a_k \in A, \nexists L_j \in T_i : C_l, C_k \in L_j \text{ and } C_l \text{ precedes } C_k \text{ in } L_j$$

We assume that a literal $\text{a}_\text{k}$ is defined in context and $\text{C}_\text{k} \neq \text{C}_\text{i}$. Partial orders are acyclic, therefore it cannot happen that A defeats B and B defeats A based on condition 2.

To link arguments through the mapping rules that they contain, we introduce in our framework the notion of *argumentation line*.

**Definition 8.** *A*n argumentation line $A_L$ for a literal $p_i$ is a sequence of arguments in $Args_C$, constructed in steps *as follows:*

- *In the first step add in* $A_L$ *one argument for* $p_i$.
- *In each next step, for each distinct literal* $q_j$ *labeling a leaf node of the proof trees of the arguments added in the previous step, add one argument with conclusion* $q_j$, *with the following restriction.*
- *An argument B for a literal* $q_j$ *can be added in* $A_L$ *only if there is no argument* $D \neq B$ *for* $q_j$ *already in* $A_L$.

The argument for $p_i$ added in the first step is called the head argument of $A_L$. If the number of steps required to build an $A_L$ is finite, then $A_L$ is a finite argumentation line. Infinite argumentation lines imply loops in the global knowledge base. Arguments contained in infinite argumentation lines participate in *attacks* against counter-arguments but may not be used to support the conclusion of their argumentation lines.

The notion of supported argument is meant to indicate when an argument may have an active role in proving or preventing the derivation of a conclusion.

**Definition 9.** *An argument A is supported by a set of arguments S if:*

- *every proper subargument of A is in S and*
- *there is a finite argumentation line* $A_L$ *with head A, such that every argument in* $A_L - \{A\}$ *is in S*

That an argument *A* is *undercut* by a set of arguments *S* means that we can show that some premises of *A* cannot be proved if we accept the arguments in *S*.

**Definition 10.** *A defeasible local or mapping argument A is undercut by a set of arguments S if for every argumentation line* $A_L$ *with head A: there is an argument B, such that B is supported by S, and B defeats a proper subargument of A or an argument in* $A_L - \{A\}$.

The definition of *acceptable* arguments that follows is based on the definitions given above. Intuitively, that an argument *A* is *acceptable* w.r.t. *S* means that if we accept the arguments in S as valid arguments, then we feel compelled to accept *A* as valid.

**Definition 11.** *An argument A is acceptable w.r.t. a set of arguments S if:*

1. *A is a strict local argument; or*
2. *(a) A is supported by S and*
   *(b) every argument defeating A is undercut by S*

Based on the concept of acceptable arguments, we proceed to define justified arguments and justified literals.

**Definition 12.** *Let C be a MCS.* $J_i^C$ *is defined as follows:*

- $J_0^C = \emptyset$;
- $J_{i+1}^C = \{A\ Args_C \mid A\ is\ acceptable\ w.r.t. J_i^C\}$

The set of *justified arguments* in a MCS C is $JArgs^C = \bigcup_{i=1}^{\infty} J_i^C$. A literal $p_i$ is *justified* if it is the conclusion of an argument in $JArgs^C$. That an argument *A* is justified means that is resists every reasonable refutation. That a literal $p_i$ is justified, it actually means that it is a logical consequence of *C*.

Finally, we also introduce the notion of *rejected arguments* and *rejected literals* for the characterization of conclusions that do not derive from *C*. That an argument is rejected by sets of arguments *S* and *T* means that either it is supported by arguments in *S*, which can be thought of as the set of already rejected arguments, or it cannot overcome an attack from an argument supported by *T*, which can be thought of as the set of justified arguments.

**Definition 13.** *An argument A is rejected by sets of arguments S, T when:*

1. *A is not a strict local argument, and either*
2. *(a) a proper subargument of A is in S; or*
   *(b) A is defeated by an argument supported by T; or*

*(c) for every argumentation line $A_L$ with head A there exists an argument $A' \in A_L - \{A\}$, such that either a subargument of $A'$ is in S; or $A'$ is defeated by an argument supported by T.*

Based on the definition of rejected arguments, $R_i^C$ is defined as follows:

**Definition 14.** *Let C be a MCS, and $JArgs^C$ the set of justified arguments in C. $J_i^C$ is defined as follows:*

- $R_0^C = \emptyset$;
- $R_{i+1}^C = \{ A \in Args_C \mid A \text{ is rejected by } R_i^C, JArgs^C \}$

The set of *rejected arguments* in a MCS C is $RArgs^C = \bigcup_{i=1}^{\infty} R_i^C$. A literal $p_i$ is *rejected* if there is no argument in $Args^C - RArgs^C$ that supports $p_i$. That a literal is rejected means that we are able to prove that it is not a logical consequence of *C*.

### 3.2.2 Properties of the Framework

Lemmata 1-3 describe some formal properties of the framework. Their proofs are simple modifications of the proofs included in the PhD thesis of Bikakis [16].

**Lemma 1.** *The sequences $J_i^C$ and $R_i^C(T)$ are monotonically increasing.*

**Lemma 2.** *In a Multi-Context System C:*

- *No argument is both justified and rejected.*
- *No literal is both justified and rejected.*

**Lemma 3.** *If the set of justified arguments of C, $JArgs_C$ contains two arguments with complementary conclusions, then both arguments are local arguments of the same context.*

The 4th presented lemma is an extension of [16] by partial preference relations. The following never states that it is compatible with the original work.

**Lemma 4.** *The new system given as input a total order, it provides the same result as the original approach of Bikakis et al.*

The proof is a straight forward checking of the basic definitions of the two frameworks.

Finally, we can show that when moving from total to partial trust relations, we may lose some justified conclusions. To be more specific:

**Lemma 5.** *Let C be an MCS with partial trust relation, and C' an MCS in which all trust relations are linear extension of those in C. Then* $RArgs^{C} \supseteq RArgs^{C'}$ *and* $JArgs^{C} \subseteq JArgs^{C'}$.

**Example (continued).** The arguments that are derived from $MCS\ C$ of the section 1.2.1 are depicted in Figure 3.1 along with their subarguments.

$J_0^C$ contains no arguments, while $J_1^C$ contains the strict local arguments of the system; namely $A_1', A_1'', A_{21}, A_{22}, A_4, A_5, B_3, D_6$, where $A_1'$ and $A_1''$ are the strict local subarguments of $A_1$with conclusion $\text{likesMovie}_1$ and $\text{actionMovie}_1$ respectively.

$J_2^C$ additionally contains $A_1$'s subarguments with conclusions $\text{goodWeather}_1$, $\text{freeSchedule}_1$, $\text{goodMovie}_1$ as well as $B_1$'s subargument with conclusion $\text{badWeather}_1$ and $D_1$'s subargument with conclusion $\text{badMovie}_1$. $J_2^C$ actually constitutes the set of justified arguments in C ($\text{JArgs}^C = J_2^C$), as there is no argument that can be added in the next steps of $J_i^C$.

On the other hand, $R_0^C$ ($\text{JArgs}^C$) contains no arguments, while $R_1^C$ ($\text{JArgs}^C$), which equals $\text{RArgs}^C(\text{JArgs}^C)$ contains $A_1, B_1$ and $D_1$, as each of them is defeated by a justified argument. Hence $\text{activity}_1$ and $\neg\text{activity}_1$ are rejected literals since every argument supporting them is rejected.

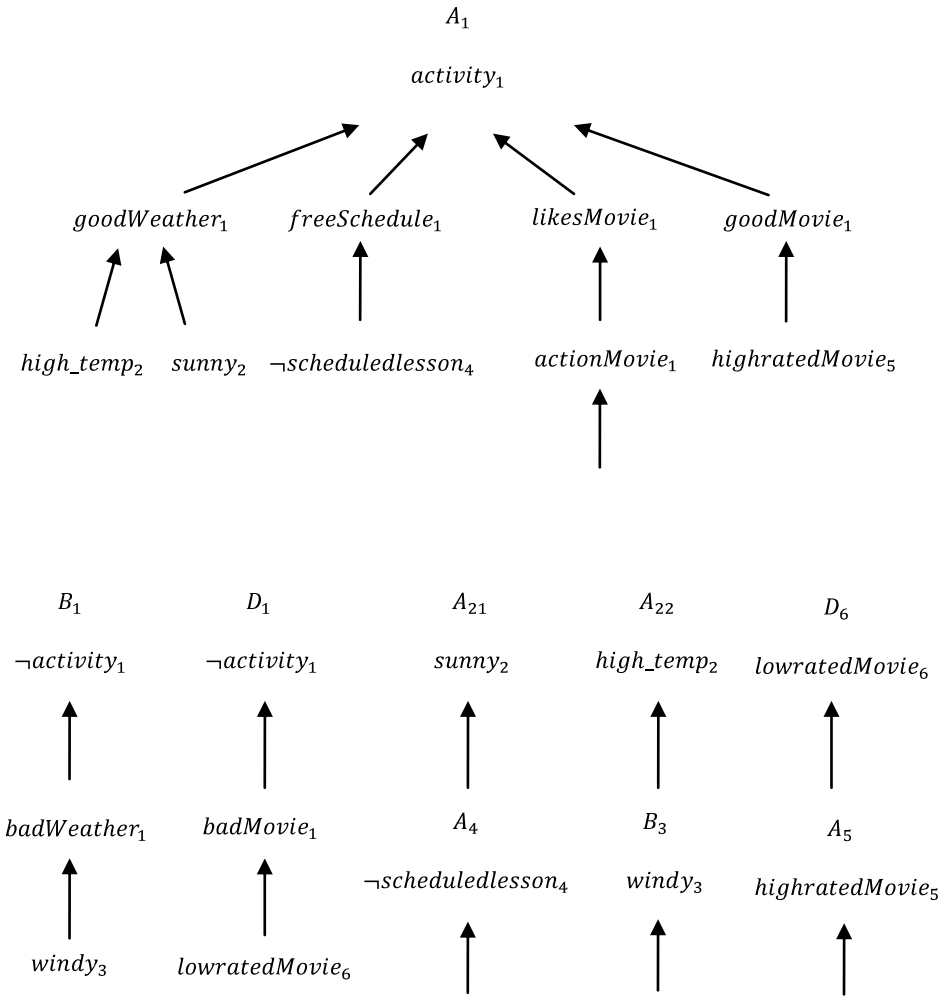



**Figure 3.1:** Arguments in the scenario.

# Chapter 4

# Distributed Algorithm for Query Evaluation

In this section we present the Distributed Algorithm for query evaluation implementing the Argumentation Semantics of the previous section, this algorithm differs from the original approach of Bikakis et al. proposed in [17] only in the *Stronger* function which implements the comparison between two sets of literals. Below, we describe the P2P_DR algorithm and give emphasis to the code of the *Stronger* function that implements the modification to the original approach. Finally, we describe the existing alternative strategies for conflict resolution based on that algorithm in order to introduce a new one more implementation oriented.

## 4.1 Algorithm Description

*P2P_DR* is a distributed algorithm for query evaluation that implements the proposed argumentation framework. The specific problem that it deals with is: *Given a MCS C, and a query about literal $p_i$ issued to context $C_i$, compute the truth value of $p_i$*. For an arbitrary literal $p_i$, *P2P_DR* returns one of the following values: (a) *true*; indicating that $p_i$ is a logical consequence of *C*; (b) *false*; indicating that $p_i$ is not a logical consequence of *C*; or (c) *undefined*; indicating that based on C, we cannot derive neither *true* nor *false* as a truth value for $p_i$.

*P2P_DR* proceeds in four main steps. In the first step, *P2P_DR* determines whether $p_i$, or its negation ~ $p_i$ are consequences of the strict local rules of $C_i$, returning *true/false* respectively as an answer for $p_i$ and terminates.

In the second step, *P2P_DR* calls *Support* to determine whether there are applicable and unblocked rules with head $p_i$. We call *applicable* those rules that for all literals in their body *P2P_DR* has computed *true* as their truth value, while *unblocked* are the rules that for all literals in their body *P2P_DR* has computed either *true* or *undefined* as their truth value. *Support* returns two data structures for $p_i$: (a) the Supportive Set of pi ($SS_{p_i}$), which is the set of foreign literals used in the most preferred (according to $T_i$) chain of applicable rules for $p_i$ ; and (b) the Blocking Set

of $p_i$ ($BS_{p_i}$), which is the set of foreign literals used in the most preferred chain of unblocked rules for $p_i$ ($BS_{p_i}$). If there is no unblocked rule for $p_i$ ($BS_{p_i} = \emptyset$), the algorithm returns *false* as an answer and terminates.

In the third step, similarly to the second, *P2P_DR* calls *Support* to compute the respective constructs for $\sim p_i$ ($SS_{\sim p_i}, BS_{\sim p_i}$).

In the last step, *P2P_DR* uses the constructs computed in the previous steps and the preference order $T_i$, to determine the truth value of $p_i$. In case there is no unblocked rule for $\sim p_i$ ($BS_{\sim p_i} = \emptyset$), or $SS_{p_i}$ is computed by *Stronger* to be *stronger* than $BS_{\sim p_i}$, *P2P_DR* returns *true* as an answer for $p_i$. That $SS_{p_i}$ is *stronger* than $BS_{p_i}$ means that the chains of applicable rules for $p_i$ involve information from contexts that are preferred by $C_i$ to the contexts that are involved in the chain of unblocked rules for $\sim p_i$. In case there is at least one applicable rule for $\sim p_i$, and $BS_{p_i}$ is *not stronger* than $SS_{\sim p_i}$, *P2P_DR* returns *false* as an answer for $p_i$. In any other case, the algorithm returns *undefined*.

The context that is called to evaluate the query for $p_i(C_i)$ returns through $Ans_{p_i}$ the truth value of the literal it is queried about. $SS_{p_i}$ *and* $BS_{p_i}$ are returned to the querying context ($C_0$) only if the two contexts (the querying and the queried one) are actually the same context. Otherwise, the empty set is assigned to both $SS_{p_i}$*and* $BS_{p_i}$ and returned to $C_0$. In this way, the size of the messages exchanged between different contexts is kept small. $Hist_{p_i}$ is a structure used by Support to detect loops in the global knowledge base.

The input parameters of P2P_DR are:

- $p_i$: *the queried literal*
- $C_0$: *the context that issues the query*
- $C_i$: *the context that defines* $p_i$
- $Hist_{p_i}$: *the list of pending queries* $([p_1, \dots, p_i])$
- $T_i$: *the preference ordering of* $C_i$

The output parameters of the algorithm are:

- $SS_{p_i}$: *a set of foreign literals of* $C_i$ *denoting the Supportive Set of* $p_i$
- $BS_{p_i}$: *a set of foreign literals of* $C_i$ *denoting the Blocking Set of* $p_i$
- $Ans_{p_i}$: *the answer returned for* $p_i$

Below, we provide the pseudocode of *P2P_DR*

**P2P_DR** $(p_i, C_0, C_i, Hist_{p_i}, T_i, SS_{p_i}, Ans_{p_i})$

call local_alg $(p_i, localAns_{p_i})$
**if** localAns = true **then**
$Ans_{p_i} \leftarrow true, SS_{p_i} \leftarrow \emptyset, BS_{p_i} \leftarrow \emptyset$
terminate

call local_alg $(\sim p_i, localAns_{p_i})$
**if** $localAns_{\sim p_i} = true$ **then**
$Ans_{p_i} \leftarrow false, SS_{p_i} \leftarrow \emptyset, BS_{p_i} \leftarrow \emptyset$
terminate

call Support $(p_i, Hist_{p_i}, T_i, \sup_{p_i}, unb_{p_i}, SS_{p_i}, BS_{p_i})$
**if** $unb_{p_i} = false$ **then**
$Ans_{p_i} \leftarrow false, SS_{p_i} \leftarrow \emptyset, BS_{p_i} \leftarrow \emptyset$
Terminate
$Hist_{\sim p_i} \leftarrow (Hist_{p_i} - \{p_i\}) \cup \{\sim p_i\}$

call Support $(\sim p_i, Hist_{\sim p_i}, T_i, \sup_{\sim p_i}, unb_{\sim p_i}, SS_{\sim p_i}, BS_{\sim p_i})$
**if** $\sup_{p_i} = true\ and\ (unb_{\sim p_i} = false\ or\ Stronger(SS_{p_i}, BS_{\sim p_i}, T_i) = SS_{p_i})$ **then**
$Ans_{p_i} \leftarrow true$
**if** $C_0 \neq C_i$ **then**
$SS_{p_i} \leftarrow \emptyset, \quad BS_{p_i} \leftarrow \emptyset$
**else if** $\sup_{\sim p_i} = true\ and\ Stronger(BS_{p_i}, SS_{\sim p_i}, T_i) \neq BS_{p_i}$ **then**
$Ans_{p_i} \leftarrow false, SS_{p_i} \leftarrow \emptyset, BS_{p_i} \leftarrow \emptyset$
**else**
$Ans_{p_i} \leftarrow undefined$

**if** $C_0 \neq C_i$ **then**
$SS_{p_i} \leftarrow \emptyset, BS_{p_i} \leftarrow \emptyset$

*local_alg* is called by *P2P_DR* to determine whether the truth value of the queried literal can be derived from the strict local rules of a context theory. We should note that, for sake of simplicity, we assume that there are no loops in the local context theories. *local_alg* returns either *true* or *false* as a local answer for the queried literal.

The algorithm parameters are:

- $p_i$: $the\ queried\ literal\ (input)$
- $localAns_{p_i}$: $the\ local\ answer\ for\ p_i\ (output)$

**Local_alg** ( $p_i, localAns_{p_i}$)
**for all** $r_i \in R^s[p_i]$ **do**
    **for all** $b_i \in body(r_i)$ **do**
        call local_alg($b_i, localAns_{b_i}$ )
    **if for all** $b_i$: $localAns_{b_i} = true$ **then**
        **return** $localAns_{p_i} = true\ and\ terminate$
**return** $localAns_{p_i} = false$

*Support* is called by P2P_DR to determine whether there are applicable and unblocked rules for $p_i$. In case there is at least one applicable rule for $p_i$, Support returns $\text{sup}_{p_i} = true$; otherwise, it returns $\text{sup}_{p_i} = false$. Similarly, $unb_{p_i} = true$ is returned when there is at least one unblocked rule for $p_i$; otherwise, $unb_{p_i} = false$.

*Support* also returns two data structures for $p_i$:

- $SS_{p_i}$: $the\ Supportive\ Set\ for\ p_i$. This is a set of literals representing the most preferred (according to $T_i$) chain of applicable rules for $p_i$.
- $BS_{p_i}$: $the\ Blocking\ Set\ for\ p_i$. This is a set of literals representing the most preferred (according to $T_i$) chain of unblocked rules for $p_i$.

To compute these structures, *Support* checks the applicability of the rules with head $p_i$, using the truth values of the literals in their body, as these are evaluated by P2P_DR. To avoid loops, before calling P2P_DR, it checks if the same query has been issued before during the running call of P2P_DR. In this case, it marks the rule with a cycle value, and proceeds with the remaining body literals. For each applicable rule $r_i$. Support builds its Supportive Set, $SS_{r_i}$ ; this is the union of the set of foreign literals contained in the body of $r_i$ with the Supportive Sets of the local literals contained in the body of the rule. Similarly, for each unblocked rule $r_i$, it computes its Blocking Set $BS_{r_i}$using the Blocking Sets of its body literals. Support computes the Supportive Set of $p_i$, $SS_{p_i}$ , as the strongest rule Supportive Set $SS_{r_i}$ ; and its Blocking Set, $BS_{p_i}$, as the strongest rule Blocking Set $BS_{r_i}$ , using the Stronger function.

The input parameters of Support are:

- $p_i$: *the queried literal*
- $Hist_{p_i}$: *the list of pending queries* $([p_1, \dots, p_i])$
- $T_i$: *the preference ordering of* $C_i$

The output parameters of Support are:

- $\sup_{p_i}$ : *which indicates whether* $p_i$ *is supported in* $C$
- $unb_{p_i}$ : *which indicates whether* $p_i$ *is unblocked in* $C$
- $SS_{p_i}$: *a set of foreign literals of* $C_i$ *denoting the Supportive Set of* $p_i$
- $BS_{p_i}$: *a set of foreign literals of* $C_i$ *denoting the Blocking Set of* $p_i$

**Support** ($p_i, Hist_{p_i}, T_i, \sup_{p_i}, unb_{p_i}, SS_{p_i}, BS_{p_i}$ )

$sup_{p_i} \leftarrow false$

$unb_{p_i} \leftarrow false$

**For all** $r_i \in R[p_i]$ **do**

$cycle(r_i) \leftarrow false$

$SS_{r_i} \leftarrow \emptyset$

$BS_{r_i} \leftarrow \emptyset$

**For all** $b_t \in body(r_i)$ **do**

**if** $b_t \in Hist_{p_i}$ **then**

$cycle(r_i) \leftarrow true$

$BS_{r_i} \leftarrow BS_{r_i} \cup \{d_t\}\{d_t \equiv b_t \; if \; b_t \notin V_i; otherwise$

$d_t is \; the \; first \; foreign \; literal \; of \; C_i \; added \; in \; Hist_{p_i} after \; b_t\}$

**else**

$Hist_{b_t} \leftarrow Hist_{p_i} \cup \{b_t\}$

call P2P_DR($b_t, C_i, C_t, Hist_{b_t}, T_t, SS_{b_t}, BS_{b_t}, Ans_{b_t}$ )

**if** $Ans_{b_t} = false$ **then**

stop and check the next rule

**else if** $Ans_{b_t} = undefined \; or \; cycle(r_i) = true$ **then**

cycle($r_i$) $\leftarrow true$

**if** $b_t \notin V_i$ **then**

$BS_{r_i} \leftarrow BS_{r_i} \cup \{ b_t\}$

**else**

$BS_{r_i} \leftarrow BS_{r_i} \cup BS_{b_t}$

**else**

**if** $b_t \notin V_i$ **then**

$BS_{r_i} \leftarrow BS_{r_i} \cup \{ b_t\}$

$SS_{r_i} \leftarrow SS_{r_i} \cup \{ b_t\}$

**else**

$BS_{r_i} \leftarrow BS_{r_i} \cup BS_{b_t}$

$SS_{r_i} \leftarrow SS_{r_i} \cup SS_{b_t}$

**if** $unb_{p_i} = false \; or \; Stronger\left(BS_{r_i}, BS_{p_i}, T_i\right) = BS_{r_i}$ **then**

$BS_{p_i} \leftarrow BS_{r_i}$

$unb_{p_i} \leftarrow true$

**if** $cycle(r_i) = false$ **then**

**if** $\sup_{p_i} = false \; or \; Stronger\left(SS_{r_i}, SS_{p_i}, T_i\right) = SS_{r_i}$ **then**

$SS_{p_i} \leftarrow SS_{r_i}$

$sup_{p_i} \leftarrow true$

The *Stronger*$(\mathrm{A}, \mathrm{B}, \mathrm{T_i})$ function computes the *strongest* between two sets of literals, $A$ and $B$ according to the preference order $T_i$. A literal $a_k$is preferred to a literal $b_l$, if there is a list $L_j$ in $T_i$such that $C_k$ and $C_l$ are both part of that list and $C_k$ precedes $C_l$. It must be noted that literals might not be comparable if there is no list in $T_i$ containing both contexts from which these literals are derived. So in case where literals of two sets are not comparable then stronger function returns *undecided* for which set is the stronger.

**Stronger (**$A, B, T_i$)

**if** $\exists b_l \in B : \forall a_k \in A, \exists L_j \in T_i : C_l, C_k \in L_j$ ***and*** $C_k$ *precedes* $C_l$ *in* $L_j$ **then**

*Stronger = A*

**else if** $\exists a_k \in A : \forall b_l \in B, \exists L_j \in T_i : C_k, C_l \in L_j$ ***and*** $C_l$ *precedes* $C_k$ *in* $L_j$ **then**

*Stronger = B*

**else**

*Stronger = None*

**return** *Stronger*

**Example (continued from section 3).** Given a query about $activity_1$, *P2P_DR* proceeds as follows. At first, it fails to compute an answer based only on $C_1$'s local theory from which it derives only $\mathrm{likesMovie_1}$ using strict local rules $\mathrm{r_{14}^l}$ and $\mathrm{r_{15}^l}$. Next step is to use mapping rules $\mathrm{r_{16}^m}, \mathrm{r_{18}^m}$ and $\mathrm{r_{19}^m}$ to compute an answer for $\mathrm{goodWeather_1}, \mathrm{freeSchedule_1}$ and $\mathrm{goodMovie_1}$, based on strict local rules of $\mathrm{C_2}, \mathrm{C_4}$ and $\mathrm{C_5}$ respectively. These rules are applicable and support $\mathrm{activity_1}$, as are rules $\mathrm{r_{12}^d}$ and $\mathrm{r_{13}^d}$ supporting the opposite $\neg\mathrm{activity_1}$. Their respective computed sets are the following: $\mathrm{SS_{activity}} = \{\ \mathrm{high_{temp_2}}, \mathrm{sunny_2}, \neg\mathrm{scheduledlesson_4}, \mathrm{highratedMovie_5}\}$ and the $\mathrm{BS_{\sim activity}} = \{\mathrm{windy_3}, \mathrm{lowratedMovie_6}\}$.

Based on $\mathrm{T_1}$, *Stronger* returns *none* and eventually *P2P_DR* returns false for $activity_1$. The same result would occur for a query about $\neg\mathrm{activity_1}$.

## 4.2 Properties of the Algorithm

Below, we describe some formal properties of *P2P_DR* regarding its termination, soundness and completeness w.r.t. the argumentation framework, communication and computational load. Here, we give an overview of the results.

Prop. 1 is a consequence of the cycle detection process within the algorithm.

**Proposition 1.** (Termination) *The algorithm is guaranteed to terminate returning either a positive or a negative answer for the queried literal.*

Prop. 2 associates the answers produced by *P2P_DR* with the concepts of justified and rejected arguments.

**Proposition 2.** *For a Multi-Context System C and a literal $p_i$ in C, P2_ DR returns:*

1. $\text{Ans}_{p_i} = \ true\ iff\ p_i\ is\ justified$

2. $\text{Ans}_{p_i} = false\ iff\ p_i is\ rejected\ by\ JArgs^C$

3. $\text{Ans}_{p_i} = \ undefined\ iff\ p_i is\ neither\ justified\ nor\ rejected\ by\ JArgs^C$

Prop. 3 is consequence of two states that we retain for each context, keeping track of the incoming and outgoing queries of the context. The worst case is that all contexts have defined mappings that contain literals from all other contexts, and the evaluation of the query involves all mappings defined in the system.

**Proposition 3.** (Number of Messages) *The total number of messages exchanged between the system contexts for the evaluation of a query is $O(n^3\ x\ n_l^2)$, where n stands for the total number of system contexts, and $n_l$ stands for the number of literals a context may define.*

The number of exchanged messages remains the same as in the original approach using total ordering preference information, since the only difference partial order incorporates is in the *Stronger* function, which compares Supportive Sets and is irrelevant to the intermediary queries that might result in the evaluation process. The proof for the original approach is located in the appendix of [16]

As far as computational complexity is concerned it is related with the total basic calculations that a context has to perform. In general it depends on the number of the existing rules in a context and the complexity of the Stronger function. Therefore in the total ordering case, we have O(r x n), where n is the total number of literals of the system. O(n) is the complexity of finding the weakest literal in every set. Using partial order, since there is no notion of the weakest literal, there would be in the worst case a comparison of every combination of two literals (one of each of the two sets), resulting in Stronger's complexity $O(n^2)$ and total computational complexity $O(r \times n^2)$.

## 4.3 Alternative Strategies

The algorithm described in chapter 4.1 resolves conflicts that arise when mutually inconsistent information is imported from different contexts based on the rank of these contexts in the preference ordering of the context that imports this information. This chapter briefly describes three existing alternative strategies for conflict resolution (Strict-Weak Answers, Propagating Mapping Sets and Complex Mapping Sets), which differ in the type and extent of context information that is used to evaluate the quality of the imported knowledge. The intuition behind these strategies is that imported knowledge should be evaluated not only based on their "source" - the context that the knowledge is imported by - but also on the way that the "source" acquired this knowledge. The strategies along with their main features and differences follow and at the end we present a new variation for CDL.

**Strict-Weak Answers.** The distinct feature of Strict-Weak Answers, compared to the "base" strategy implemented by P2P_DR is that imported knowledge is not only evaluated based on its "source" (the context that it is imported by), but also based on whether the "source" derived this knowledge strictly-based on its strict local rules.

**Propagating Mapping Sets.** This strategy goes one step further from Strict-Weak Answers, requiring the "source" to return information about the identity of other contexts that are involved in the derivation of the imported knowledge, and evaluates

it based on the ranks of these contexts in the preference ordering of the context that imports this knowledge.

**Complex Mapping Sets.** In contrast with the previous strategy this one has the distinct feature that the most preferred between two or more reasoning chains is not determined by the queried context, but by the context that imports the knowledge. This approach is the richest w.r.t. the extent of context knowledge that it exploits, but also the one with the highest computational complexity.

The incorporation of partial order in these algorithm variations can be done similarly to the modified P2P_DR. The following strategy takes into account implementation issues that arise from the limited computational capabilities of mobile devices and wireless communication response times which are significantly slower compared to desktop computers.

**Preference Based Query.** This strategy also follows the "base" algorithm (P2P_DR) with the difference that when a context queries one another, it also sends along his preference-trust ordering (T) over the rest of the MCS's contexts. This is done so as the answer to be evaluated based on the *source's preferences*. This way the Trust is propagated along to the peers involved and the answer returned does not need any further process in the end by the context in which the query originated, since its more preferred sources (and consequently justifications) are used for the evaluation process.

As a consequence of the original context preferences being propagated to the contexts involved in a query answer, the message size actually increases since it contains the extra information of preference; but it actually has minimal effect on network message transportation time.

Summarizing, in case of missing preference information the algorithm many times will result in answering "*undefined*", through a quicker process that terminates when the first non comparable contexts and literals are encountered (by Stronger) in attempting to answer a query. Therefore in practice the number of messages exchanged and network traffic are kept to a minimum.

# Chapter 5

# Contextual Defeasible Logic System

In this section, we outline the architecture of the Mobile Contextual Defeasible Logic (CDL) System, analyze the specifics of our generic implementation, and discuss the choices and associated motivation of the technologies adopted.

## 5.1 Architecture Overview

Figure 5.1 depicts a layered overview of the system architecture. Starting from the bottom and moving upwards, first is the Reasoning and Inference layer where the reasoning is performed, it is based on Prolog and in the same layer Contextual Defeasible Logic is implemented. Next comes the Communications layer, which implements the Mobile CDL Service, a protocol for handling all incoming or outgoing communication. Next up lies the Profile and Knowledge Management layer, this is where the User Profile and Knowledge Base are stored and managed. Finally, on top, is the Application layer, which orchestrates all the underlying component interactions and provides the user interface.

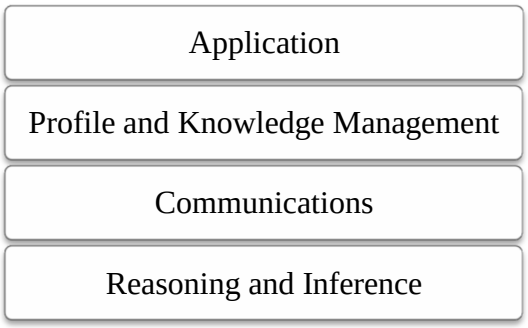



**Figure 5.1**. CDL Layered Architecture

## 5.2 Component Analysis and Implementation

All layers of the CDL System are based on the Java 2 Micro edition (J2ME) programming language. The main reasons for this choice are that it is reasonably fast,

it provides useful APIs and it is able to run on any mobile device that features a Java Virtual Machine (nowadays the vast majority of cell phones, PDAs, set top boxes etc). Moreover, it is easy to translate component code into J2SE in order to support more devices in the future.

Below we proceed to a top down description and analysis of the components comprising the Contextual Defeasible Logic System.

### 5.2.1 Application

The Application handles all the component interactions that occur on the layers underneath it and also provides the forms for the user to interface with all supported operations of CDL.

In order to run the CDL Application, a user must load his Profile and store it locally (on the device). This is done through a form in the CDL Mobile Application where user inputs an Internet URL pointing to e.g. a web page with his public Profile. After the Profile is stored the system can be started: First comes the initialization of the Reasoning and Inference Component, this is where the metaprogram, the stored user Profile, as well as any other locally available Knowledge is loaded into. Next, CDL communication protocols must be started; these are handled by the Communications component. This stage involves the startup of Wi-Fi, Bluetooth and SMS capabilities of the CDL System, provided they are supported by the given device hardware.

After a proper startup, the CDL System can accept and issue queries. The answers for these queries are evaluated based on the *distributed algorithm* described in the previous section. User can issue queries using the CDL Application forms supporting query operations: through the Internet using Wi-Fi or 3G-GPRS, over Bluetooth or even via SMS. The answer is received from the same channel used for issuing the query. Figure 5.2 shows a query operation using various peers and communication technologies.

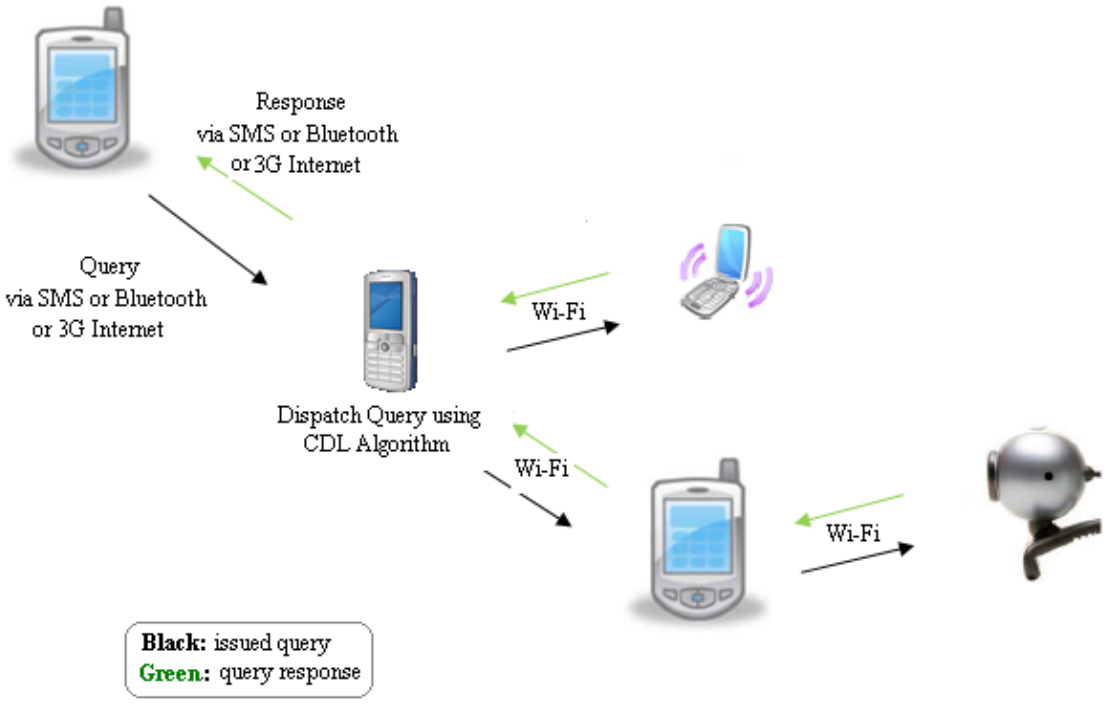



**Figure 5.2:** Technologies involved in a Query Operation

### 5.2.2 Profile and Knowledge Management

Profile and Knowledge Management layer is where the User Profile and Knowledge Base are loaded, stored and updated. Profile and Theory loading can be done locally through forms provided by the application or it can be loaded from a remote location e.g. it can be published on a web page, given that it follows a certain format (Defeasible Logic Format). This facilitates sharing of Knowledge Bases and publishing public Profiles. Moreover it is practically easier to write and maintain a Theory or a Profile on a computer keyboard rather than on a small device.

Figure 5.3 shows an example of Profile published on a web page containing facts, local defeasible rules, priorities among them, mapping rules, a preference order over peers (T) as well as information about known peers such as their name, an IP address for contacting them and of course the type of queries they answer – what they know about (see last subsection). It uses an HTML form and a hidden input element so as to be viewable only by machine since it is not in a useful format for humans

(e.g. described in physical language). This HTML Fragment represents a published profile at e.g. *http://www.csd.uoc.gr/~cpapath/publicProfileA.html*.

This profile can be loaded by filling in the appropriate URL in an input form at the CDL Application, which will use the communication component in order to retrieve it and parse it accordingly before feeding it to the Reasoning and Inference component.

```
<form>
<input type="hidden" name="profile"
value=">>>>><br>__prefs__p2__p1__p3__defProfile__fact(human(a)).
fact(female(a)).
fact(human(b)).
fact(male(b)).
defeasible(r1,man(X), [human(X)]).
defeasible(r2, oxi(man(X)), [human(X), female(X)]).
sup(r2,r1).__remMappings__defeasible(rm1, fly(penguin),
[bird(penguin), wings(penguin)] ).
defeasible( rm2, lessonInProgressAt(X),[peopleAtRoom(X),
scheduledLecture(X)]).
__peerRecs__p1__wings--socket://139.91.183.20:5491
p2__bird--socket://139.91.183.20:5491
p3__lessonInProgress--socket://139.91.183.11:5491
<br><<<<<" />
</form>
```

**Figure 5.3:** Sample user Profile published on web page

The *profile* contains various types of information and is formatted in a special way and cannot be loaded immediately into the Reasoning Engine. As a consequence, it first has to broken down into subsections, some of which must be further processed. As is depicted in Figure 5.3, if we throw away the html code encapsulating the profile (before the first "<br>" and after the last "<br>"), then the rest is the actual profile.

- The first subsection is the one containing the peer preferences – (trust) and it is formatted as follows: It always starts with the prefix "__prefs__", and continues with the names of the peers in descending order of preference. Each name is followed by the "__" suffix used for parsing. Peer names are concatenated one after the other as the example profile "__prefs__p2__p1__p3__". After parsing this subsection, a data structure is created containing the peer preferences.

- The start of the second subsection (which is also marks the end of the previous one) is signaled by the "__defprofile__" prefix which reveals the kind of information it contains. This is the core defeasible profile which can be immediately loaded into the reasoning engine and is used for the local reasoning. It contains the defeasible theory described using facts, strict and defeasible rules and priorities in the typical DR-Prolog syntax following the model presented in section 3.

- The prefix *"*__remMappings__" is the start of the third subsection containing the mapping-bridge rules. This section is important for the peer to peer algorithm because it bridges the peers for the information to flow between them. Their format remains the same as the defeasible rules, but they are subdue to a process where they are stored in a data structure and along with the preferences data structure (from the parsing of the first subsection), they are used for the peer to peer reasoning.

- Last subsection of the profile is the description of the peers starting with the prefix "__peerRecs__". Immediately after, follow the peer records in the format: name, suffix "__", the kind of query they can answer in an abstract way (see last section), suffix "--"and finally the socket type://IP address: port at which the CDL service is running and through which the given peer can be contacted. E.g. __peerRecs__p1__wings--socket://139.91.183.11:5491. These records with peer information are stored also in another data structure for the needs of the peer to peer reasoning algorithm.

All the above data structures are created once when the profile is loaded and are subdue to error checking during the parsing. The defeasible profile from the second subsection is used intact for the local reasoning as opposed to the rest which are used by the peer to peer algorithm for the distributed part of the reasoning.

### 5.2.3 Communications

Communications layer, implements the Mobile CDL Service, which is actually a communication protocol for peers using the CDL Application. It uses networking

capabilities provided by mobile devices so as to enable various communication channels between them. Using the communications layer and through the internet, Theories and Profiles can be loaded remotely.

Communication between nodes using the CDL System is achieved through messages sent over the following communication channels:

a) the Internet through Wi-Fi, GPRS or 3G
b) P2P connections based on Bluetooth and
c) use of GSM cellular network to send and receive SMSs.

Consequently, when a node wants to communicate, a channel must be selected and established so as to be able to send messages. If it is a known, meaning registered in the profile peer then the choice is between Internet IP Address and SMS Number depending on what address is available. Otherwise Bluetooth can be used for querying CDL Systems in its proximity (6-10 meters depending hardware). In the first case, when an Internet IP Address is available, an Internet Socket is set up and through it, the message containing the query and its response are exchanged, after that the socket is destroyed. In case of an SMS number the same message that would have been sent over Internet Socket is now encapsulated in an SMS and the answer is received also by SMS, all this provided that a cellular network is available. If we do not desire to reach a specific peer and we want to issue a query to all available CDL Systems in proximity e.g. for 'advertising' - broadcasting something in the form of a query, then a Peer to Peer Socket is established (and destroyed) round robin with all present devices supporting Bluetooth and implementing CDL protocol. Same as previously, a query message is used for communicating. This message actually contains text and has a certain format a fixed part and after that it contains peer preferences after that another fixed part followed by the actual query. Figure 5.4 depicts the CDL query message format. Similarly is specified a message for a query answer as well as the capsule of the SMS message that also contains cellular number of destination and sender.

```
__peerPrefs__p2_p3_p1__qry__defeasibly(father(john))
```

**Figure 5.4:** Format of CDL query message

For the communication component, we used a custom-built library based on the J2ME network packages. The message exchanging protocol in our library is simple and straightforward and combines APIs for Bluetooth, wi-fi and sms messaging to achieve communication between peers. A peer can be any networked entity that implements one or more of the CDL protocols; it can be a sensor, phone, PDA, or a PC. Each peer operates independently from all other peers and is identified by a Peer_Name, which is actually an ID on the discovery Catalog for CDL services. This catalog is for "tracking" the IP address of devices which have not a standard IP, as a result of changing networks and location. These kinds of peers can publish an IP network address under their Peer_Name with the CDL protocols. Upon starting up (or restarting due to network change or generally when an error occurs) its wi-fi server the device updates a central directory of Peer_Names and IPs, which is used by all devices to update (every time_interval) their PeerRecords Structure described in the previous sub-section.

Libraries such as JXTA [43] or agent-based frameworks would be inefficient due to the complexity in configuring especially on mobile devices. JXTA usually uses also connections through intermediary peers in order to connect to edge peers. Moreover the XML messages, their construction and parsing that JXTA and generally agent-based frameworks use is still a high load operation for most mobile devices. Also Bluetooth and GSM protocols (e.g. sms messaging) are supported as interfaces by JXTA but these interfaces must be implemented from scratch.

In summary, our network library is customized for the CDL System and the communication component interleaves the provided APIs with server threads and also implements a discovery service. However it uses an abstract network interface, which is the one that in the end the distributed algorithm uses, therefore it could be replaced by any other peer communication library that might implement this interface.

### 5.2.4 Reasoning and Inference

This component is the heart of the CDL System where the reasoning is performed. Our system relies on Prolog for basic reasoning tasks, and generally any Prolog system implemented in J2ME can be used for this purpose. This means that given a

prolog implementation in J2ME that supports a certain API for handling its reasoning engine, with a simple implementation of some programming interfaces needed by the CDL System it can be used as an alternative engine for our System.

In our current implementation we adopted TuProlog [31], as it is fast and provides an easy to use API for integration. Here, in the same layer, is implemented the Contextual Defeasible Logic, an implementation of the distributed query evaluation algorithms described in [19]. The latter is based on ideas of the logic metaprogram that simulates the proof theoretic semantics of Defeasible Logic [8].

Local reasoning is performed using this metaprogram and is done at the above mentioned prolog engine. The CDL query algorithm though, is implemented again in J2ME with some minor changes (mainly for the communication needs) and uses the data structures created during the loading of the profile. Also it coordinates local reasoning (queries to local defeasible theory) with distributed (queries including mapping rules). Using a standard programming language as Java, permits us to bridge prolog based local reasoning with distributed reasoning involving network sockets and other technical issues (such as triggering events, changing Application level interfaces etc.) that cannot be overcome using only prolog. In summary, we implemented in Java an algorithm based on [19] which breaks down a query that is answered through mapping rules. Then it dispatches those sub-queries, according to the preferences tied to the given query, to the appropriate peers which in turn use a prolog engine for querying their local theory. Finally, when there is a result, the algorithm is used again to report back to the entity-peer where the distributed query originated in order for it to synthesize the final answer.

Figure 5.5 depicts the main operations of each component we described above as sub-modules and in which layer of the general architecture they fall into.

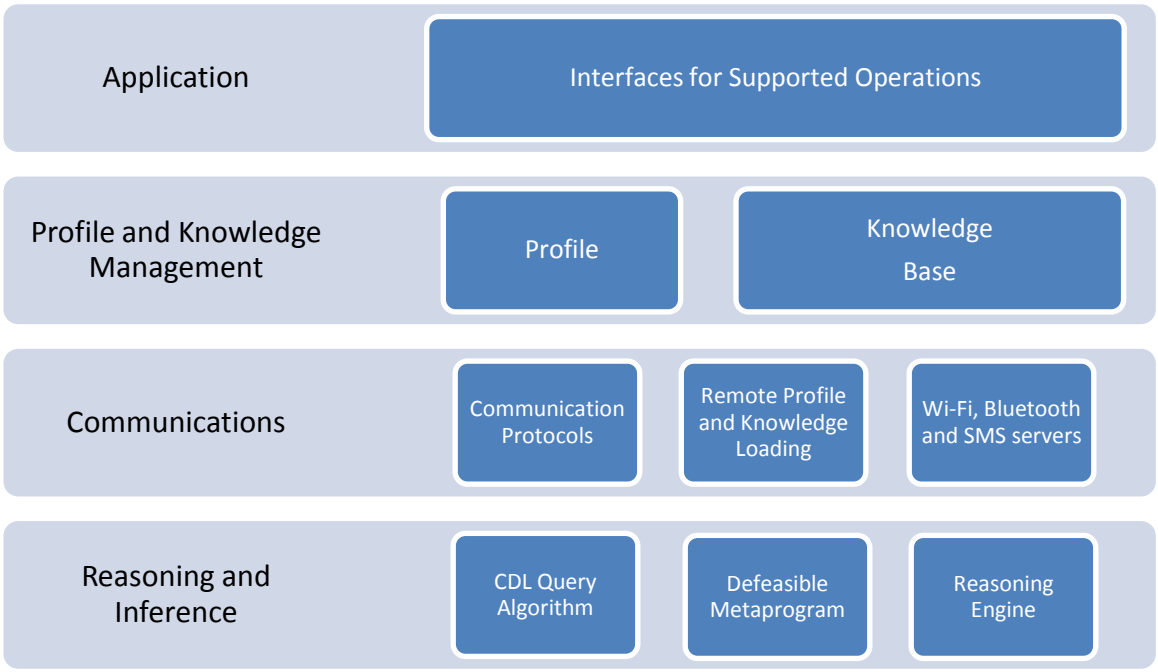



**Figure 5.5:** Main Operations as Architectural Sub-Modules

## 5.3 Extensions and Optimizations

An extension of the CDL query algorithm was implemented in order to permit reasoning using more than one peer that may provide the same kind of information. This is achieved through another version of the algorithm where a mapping rule is abstracted from just one specific peer, for example, the simple mapping rule:

$$\text{highTemp}_{\text{meteo.gr}} \Longrightarrow \text{goodWeather}_{\text{me}}$$

implies that for *me*, the querying peer, good weather is when I ask the peer *meteo.gr* for highTemp and it answers *me* yes. Now in the extended version, the rule is transformed as follows:

$$\text{highTemp} \Longrightarrow \text{goodWeather}$$

Now this rule seems to be no different to the defeasible ones and that is the truth. But since it is located in the mapping rules section of the profile, during loading it is stored in the mapping rule data structure and it is not integrated in the local defeasible theory. Therefore it is treated different that the defeasible ones during the algorithm execution.

**Algorithm execution**. As a consequence, when a query comes through and asks about goodWeather, the algorithm sees that goodWeather is a mapping rule according to the mapping rule data structure and therefore checks the data structure holding the peers records for peers answering that *kind* of query. If there is more than one entry e.g. meteo.gr, yahooWeather etc. then these are asked according to the *peer preferences* of the demanding peer that accompany the query. In fact, less preferred peers are asked only when peers answering same *kind* of query preceding them in the list are unreachable (e.g. peer is down). This is an optimization since it reduces the actual number of the messages exchanged and saves us the computation time to decide afterwards which peer to trust in case of conflicting responses.

The abstraction of the *kind of query answered* is used for generality because queries and mapping rules in reality might be more specific than the above example. E.g. goodWeather(country, city) and highTemp(country, city) or goodWeather(country,city, which_day_and_time) and highTemp(country, city, which_day_and_time) etc. Therefore the search in the structures is done taking into consideration only the outer predicate: for instance goodWeather and highTemp. Once it is decided by the algorithm which peers are to be queried then the whole query is dispatched and from then it is up to the queried node to support the given query. For instance if meteo.gr has in its Knowledge a rule with head highTemp(X, Y) or highTemp(X, Y, Z) then it would fire, that is actually the assumption when the mapping rules are added in the profile by someone. Finally someone could use a peer record more specific as far as query *kind* is concerned e.g. goodWeather(greece, heraklion, now) if this is known to be supported by that peer.

The figures 5.6 and 5.7 that follow are screenshots of the application interface of the CDL System and demonstrate query operations by showing the trace that occurs in the background from the query execution and the exchanged communication messages between the peers involved.

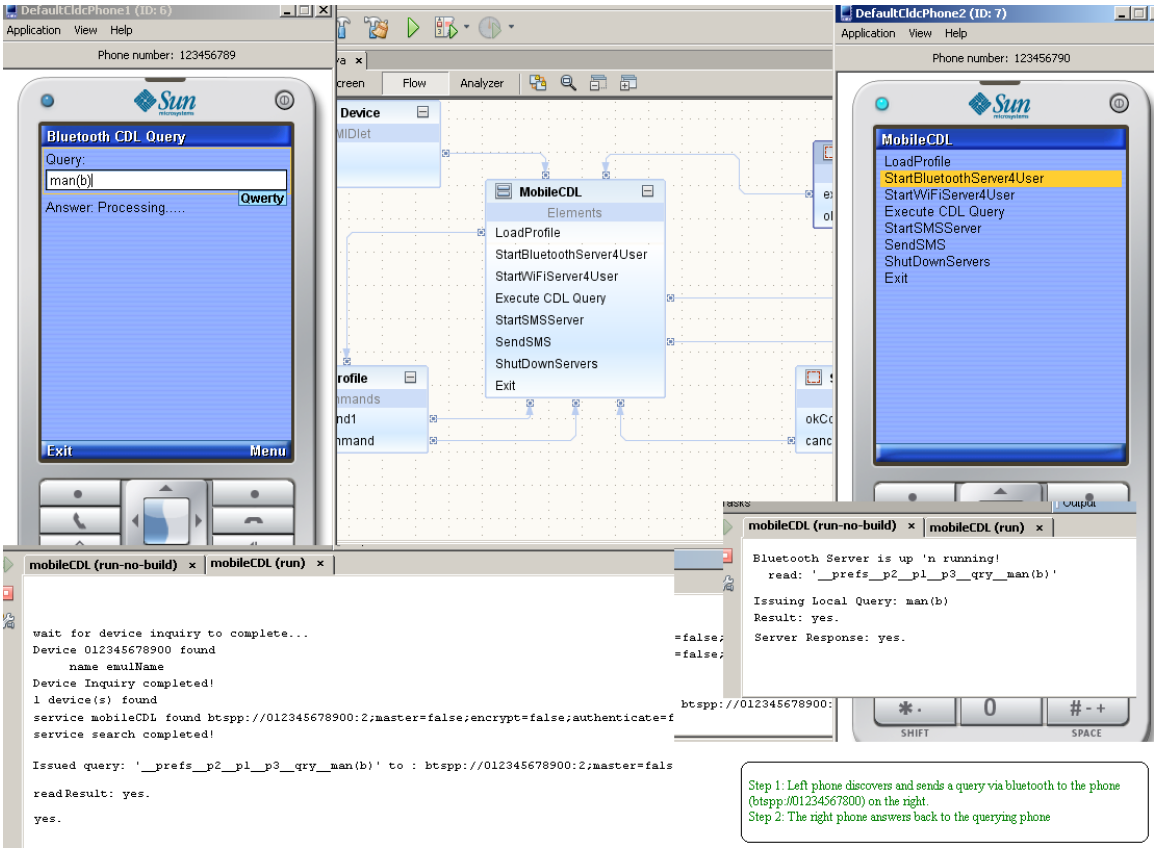



**Figure 5.6:** Query interaction using Bluetooth.

The first case, figure 5.6 involves two devices, one makes a Bluetooth query to the other, the receiving peer can provide an answer by using only its local theory and reports it back through the same communication channel.

The second case in figure 5.7 is more complex as it involves 3 peers. First peer sends query using SMS, the receiver of the SMS matches the query to a head of a mapping rule and based on it, it breaks down the original query into two sub-queries which in turn, are dispatched to the appropriate peers that answer that kind of queries. More specifically, the first sub-query is sent to the third peer and the second can be answered locally. Finally the second peer synthesizes the results from the two sub-queries and answers using SMS to the peer where the query first originated.

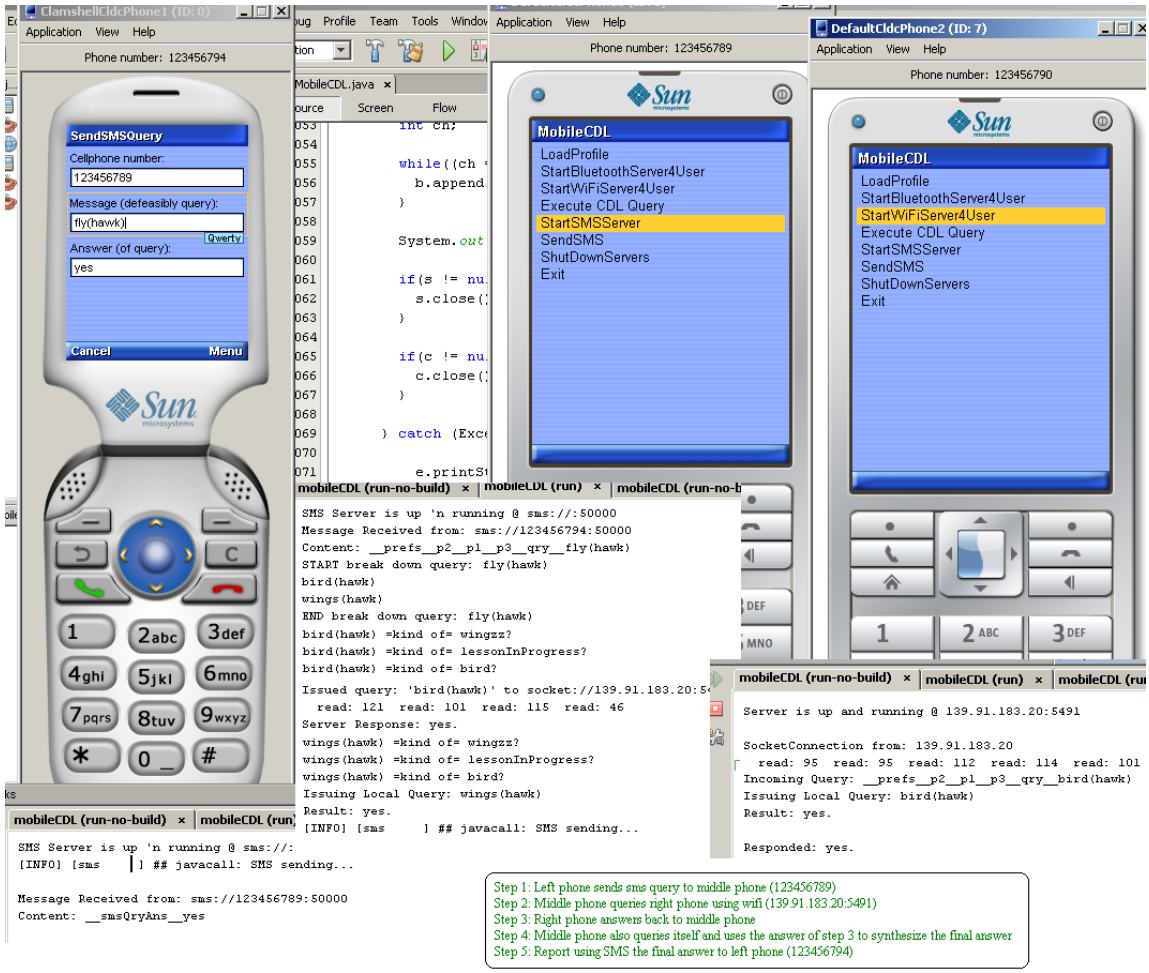



**Figure 5.7:** Traces & Profiles of cell phone query interaction through sms .

# Chapter 6

# Experimental Evaluation

In this section we performed various experiments using the CDL System. Two types of tests were conducted, the first was about verifying the feasibility of the approach on real applications based on the scenarios from the introduction, the second was about measuring the performance based on formalized metrics (query timings, theory loading, network communication etc.). After each experiment we present and discuss the results. The goal of the experiments was twofold: (a) to test whether the actual computing time for defeasible reasoning on mobile devices is acceptable, and (b) to determine the communication cost for various network types, using realistic test theories in a real environment.

## 6.1 Technical Configuration of Scenarios

The basic idea behind most of scenarios can be summarized as follows: *"When I need to know something, first ask an already known specialized source. Alternatively, if I don't know of such a source or I cannot get in touch with it, ask someone around me. In exchange, when I receive new information that may be of interest to others, I should pass it on to them."*

It is undeniable that a mobile phone is an inseparable part of almost any person nowadays; moreover, there is a clear tendency for cell phones to be more than just phones, their computing power increases and offer full wireless networking capabilities combined with GPS and other sensors. Therefore, mobile phones are at the heart of both scenarios.

**Context-Aware mobile phone in Ambient Classroom.** This scenario has been implemented at the ICS-FORTH "AmI SandBox" [69]. The SandBox consists of three rooms equipped with state of the art AmI hardware, made available through a middleware infrastructure. It is called a sandbox as it is the predecessor of a full an intelligent building planned to be finished early 2011.

This scenario involves 4 contexts (devices):

1. Dr. Amber's mobile phone; it is WiFi enabled, it carries Dr. Amber's profile, and the Mobile CDL Application is installed on it;
2. his laptop, running a Java program used to view and edit his daily schedule;
3. a computer, managing classroom cameras, with two Java programs implementing the classroom computer manager and the localization service for monitoring Dr. Amber's cell phone
4. a Radio Frequency IDentification (RFID) set.

All contexts are connected to the ICS-FORTH network.

*RFID*. To monitor the location of Dr. Amber we attached an RFID tag to his mobile phone. The RFID reader is connected to a computer implementing the fourth context (*d*), updates the location knowledge and acts also as a server by responding to queries for *currentLocation*(*X*). Specifically, it responds with either *X*='a*mi*' if the RFID reader inside the classroom reads Dr. Amber's tag, or *X*='*unknown*' otherwise.

*Classroom manager*. It responds to queries concerning the number of persons present in the classroom deduced by the AmI SandBox camera service. Queries are fixed to: *headCount*(*X*,*Location*), where *Location* is instantiated for the needs of the scenario with '*ami*'.

*Laptop*. The laptop Java program implements a calendar and responds to queries of the following fixed form: *classtime*(*Location*), where *Location* is instantiated with '*ami*'. Assuming there is such an entry, if the current time and date is in between begin and end time of that entry, the response is *true*, otherwise *false*.

*Mobile phone*. It runs Mobile CDL and contains a profile with the rules dictating when the phone should ring. This profile is a defeasible theory expressed in Prolog syntax, and is loaded to the Prolog engine along with the CDL metaprogram. The knowledge of the other contexts is assembled and transformed into defeasible theory facts.

| |
|---|
| headCount(NumOfPersons,Location), NumOfPersons < 3 → ~classactivity(Location). |
| **r0:** ~classactivity('ami') ⟹ ring. |
| **r1:** classtime('ami'), currentLocation('ami') ⟹ ~ring. |
| **r2:** currentLocation('unknown') ⟹ ring. |
| **r0 > r1.** |
| **r0 > r2.** |
| **r1 > r2.** |

**Table 6.1:** Dr. Amber's profile

Finally, an SMS server runs in a background thread of the Mobile CDL application to listen for incoming sms messages. This is achieved using a J2ME API for sending and receiving SMSs on certain ports, in combination with another API that we use to wake the application upon receiving SMS on a port it is registered to listen to.

Putting it all together, receiving an SMS acts as the trigger for a query execution in order to decide whether the cell phone should ring or just vibrate and flash its screen (silent mode).

Dr Amber's profile is composed of the theory depicted in Table 6.1, and a set of facts collected from other contexts (devices). In this case, the facts are: *currentLocation*('*ami*'), *headcount*(1,'*ami*') and *classtime*('*ami*') for the localization, camera and calendar contexts respectively. Due to rule r0's superiority over r1, the ring decision is reached.

**Social Networking Scenario.** Due to the nature of social networks, this scenario involves pre-known contexts e.g. server nodes, which are assumed to be at a fixed location with static IP address, and others that cannot be known beforehand such as user cell phones or servers without internet connection.

This scenario takes place at ICS-FORTH, downtown and the University at the same time (all located in different parts of the city). For this scenario, the only known context is the University of Crete web server, used for storing user profiles and making them available online, two laptops equipped with Bluetooth adapters pose as

the University front desk and ICS Lobby servers, which in this scenario were not considered pre-known.

With the exception of cell phones running the Mobile CDL Application, all servers are using Java programs for the implementation of contexts, similar to those described in the previous scenario. Overall, every system node must implement the appropriate CDL communication protocols.

Bluetooth server nodes advertise information of local significance. Such information may refer to an announcement of a lesson cancellation from the University Front Desk, or a warning issued by ICS-FORTH parking security for visitors and personnel passing by its lobby. Finally, users have their profile and preferences loaded on their Bluetooth enabled mobile phones or PDAs, so that the Mobile CDL can cross reference it with information from Bluetooth servers, other users, or any known servers registered in the profile.

Below, we describe the specifics of profiles used (abstracting from issues of daytime to keep the presentation simpler). The basic user profile is shown in Table 6.2. Specific additions for the users are: user A also has *workon*(’*17-12-2009*’), user B has a public profile on the university web server. He advertises it with Bluetooth: *publicProfile*(‘*http://www.csd.uoc.gr/~userB*’,’*B*’). Both users, A and B have *interest*(‘*tennis*’). Profiles also have friend entries: users A and C have *friend*(‘*university*’, ’*B*’, ’*697123456*’), and B has *friend*(‘*university*’, ’*A*’, ’*697123458*’) and *friend*(‘*university*’, ’*C*’, ’*697123569*’).

interest('semantic web').
registered('CS585').
**r0:** friend(‘university’,Y,Z) → smsNotification(friend(X,Y,Z)).
**r1**: interest(X) ⟹ notify_user(event(X,Y,Z)).
**r2**: workon(Z) ⟹ ~(notify_user(event(X,Y,Z))).
**r2**>**r1**.
**r3**: registered(X) ⟹ notify_user(announcement(X,Y)).
**r4**: birthday(X,Y,Z), currentDay(X,Y,K) → notify_user(birthday(X,Y,Z)).
**r5:**friend(K,Y,L) → registerProfile(publicProfile(X,Y)).

**Table 6.2:** Common Part of User Profile

The ICS-FORTH Lobby server contains the following event-related facts: *event*('*semantic web*', '*lecture on semantic web at ICS on*', '*18-12-2009*') and *event*('*tennis*', '*ICS tennis tournament*', '*17-12-2009*'). The CSD Front Desk server contains: *announcement*('*CS585*', '*cancellation of today's lesson lecture*'). Finally, the university web server profile pages contain birthday information.

The Mobile CDL Application has been configured to execute different queries depending on the advertised knowledge. When an announcement, event or birthday information is received then the *notify_user* query is performed. If it is about a public profile, then the *registerProfile* query is executed, and if *true* is returned, then Java is used for adding it to the profile. Also, after adding a new public profile, and once a day if Internet is available, through Java, registered public profiles are checked for birthday entries and cross-referenced with the current day being updated every day in the user cell phone profiles (e.g. *currentDay*('*12*','*12*','*2009*')), thereby informing its user.

Next, we describe the workflow of the concrete scenario. Initially, all three users were drinking coffee together downtown. At that time, B's mobile phone advertises his public profile and triggers a query execution: *registerProfile*(*publicProfile*('…','*userB*') to discovered cell phones around. A and C are friends of B's, so their mobiles register his public profile. This triggers the birthday query execution, and A and C are notified of B's birthday. Then A goes to university, C goes to ICS-FORTH while B remains downtown. After a while, the University Front Desk attempts to inform every Bluetooth device in range of class cancellation. It discovers user A and triggers the execution for two queries on his cell phone: (a) *notify_user*(*announcement*(*X*,*Y*)) to decide whether B will be informed of this announcement, and (b) *smsNotification*(*friend*(*X*,*Y*,*Z*)) to forward the news to B's university friends via SMS, regardless of the result of query (a). After B receives the SMS, similar reasoning results in notifying his friend C as well. Finally and a moment later, user C passes by the ICS lobby, where the new announcements regarding lecture and tournament are received and passed on to B and C in the same way.

## 6.2 Scenario based evaluation results

**Scenario 1 results**. The time required for communication between the mobile phone and a server was ranging between 80 and 100 milliseconds and since the message size was up to 10 bytes the actual overhead was mainly the round trip time. The scenario was also simulated with a cell phone emulator on a laptop and the timings were about 15ms; obviously, cell phones are slower handling wifi connections but still within acceptable limits.

The initialization of the reasoning engine and the theory loading (occurring only once) required about 600ms; afterwards there is a small overhead of about 20ms for minor updates to the theory when needed. Finally, while the query execution was completed in about 150ms, the total time for the scenario was about 800ms since it included some other tasks e.g. measuring and printing timings, or 320ms if the reasoning engine was already initialized, which is the standard case. These numbers are a rounded average of 10 executions.

For the timings we tested 3 devices: two regular phones and a PDA, all wifi enabled but rather old (3 to 4 years old). The oldest mobile phone was slower about 3 times (on all aspects) than the other cell phone and 5 times slower than the PDA, and thus is considered inadequate. For the cell phone simulator on the laptop the overall time needed was about 200ms.

**Scenario 2 Results**. This scenario involves Bluetooth discovery operation and SMS communication between nodes of both taking up time that might vary highly depending on the number of devices present and GSM network respectively.

Apart from the inevitable discovery time, the Bluetooth communication cost is practically better in contrast to those of the previous scenario that were using wifi. This is natural, since the data sent is text and it involves only a direct connection of the two nodes, as opposed to at least one more node e.g. a wireless router that is needed for connection to a W-iFi network.

For the two back to back queries sent by the Bluetooth server at ICS Lobby the overall time needed was about 5ms for data transmission and around 80ms for the two queries. The Reasoning Engine is initialized during application start-up, prior to the scenario trigger. Once the SMS is received, its processing takes around 15ms and

the rest is about the same since the two queries combined are under 160chars long thereby fitting within a single SMS (the case of user B informing A with SMS).

Given an average 5 second SMS delay and an average of 10 seconds for the discovery of C's mobile phone, the information from the ICS Lobby is shared between all three users within 21 seconds (about 10085ms for user C, 10ms to send the SMS to user B, and 5095ms receiving and reasoning, 10ms for userB to send to user A an SMS, and 5095 seconds for user A to receive and reason).

## 6.3 General CDL System performance testing

In order to proceed with testing of the CDL System on a larger scale, we perform tests based mainly on simulation, due to lack of enough wi-fi enabled mobile devices. It has to be noted, that the mobile phone emulator, even if it is run on a computer, simulates the slow hardware of an actual mobile device but up to a certain point. Thus it is still much faster than an actual mobile device but not as fast as the actual computer it runs on. The networks used were real Wi-Fi networks, as were various servers that were used in our experiments.

In contrast to the scenario based evaluation, which occurred almost a year before using old generation cell phones (3 to 4 years old), for the more recent general CDL system performance testing we used a last generation cell phone and along with some system optimizations made on the way, we have an overall improved performance that is reflected by the new numbers.

At first we used an actual mobile phone to basic perform tests, the same tests were also performed using a mobile phone emulator on a 2GHz core duo laptop with 2GB RAM. Moreover, after observing the results of a small escalation of measurements on both cases, we were able to extract patterns and assume similar results for even more escalated cases. For instance (example with imaginary timings) the emulator needs 1 sec for a local query, on the real device it takes 2 sec, for two local queries the emulators needs 2 sec the actual device needs 4 sec etc. therefore we can assume that for single local queries the real device's performance would be about a half of the simulated one's. Furthermore, when there are networks involved e.g. wi-fi, for instance we see that the round trip time for message exchange only (no

reasoning involved) is 5 times slower on the device compared to the emulator e.g. 75ms vs 15ms. This is fact mentioned in the previous section and has to do mainly with the setup time of the socket. Therefore for a query that has three foreign literals and needs to ask three other peers, we can assume that the local queries involved would be half as fast and the message exchange would take up five times more that it would take on the simulator. The escalation that occurs is e.g. 1 query + 3 subqueries + 3 network round trip times = about 1*1000ms + 3*1000m + 3*15ms = 4045ms on simulation and 1 * 2000 + 3*2000 + 3*75ms = 8225ms.

Based on the above imaginary numbers, the following pattern appears to hold regularly about actual and emulated processing times: for a P2P query the actual predicted processing time for escalated cases (more foreign literals) would be double and opens up even more due to the slower network handling (for the device) and also the more complicated process of P2P query evaluation which demands more processing power by the device).

The following performance tests are mostly based on simulations, and use various metrics such as the number of contexts involved, the number of rules and literals defined by them. Based on these we use queries varying in complexity e.g. local queries (with results yes, no, undefined, which already vary in complexity), also queries involving an ever escalated number of foreign literals from different contexts etc. More detailed information about the methodology followed for the experiments, the test theories and the queries we used for the evaluation is located in the appendix.

**Profile and Knowledge Base loading.** This is the time needed for the reasoning engine to initialize and load a theory composed of facts, defeasible rules (with priorities) and mapping rules (with preference relation). Table 6.3 presents the times measured for loading defeasible theories differing in the number of facts and rules from which they are composed, first on a device and then on emulator.

| THEORY | DEVICE | EMULATOR |
|---|---|---|
| **20 facts + 10 rules** | 996 | 890 |
| **100 facts + 32 rules** | 1291 | 1093 |
| **200 facts + 64 rules** | 1707 | 1434 |
| **400 facts + 128 rules** | 2509 | 2029 |

**Table 6.3:** Knowledge Base loading time in msec

These numbers show that the mobile device is slower than the emulator and although on a small knowledge base (e.g. such as the one's used by our scenarios) the difference is about 10% percent (on a basis of 1 second), as the knowledge base is growing this opens up to 20% (on a basis of 2 seconds) and it will open up even more as long as the memory of the device is not full. This time difference for most cases is not dramatic as it occurs once during the Profile and Knowledge base loading and generally it is expected from a limited resources mobile device. Chart 6.1 is a visual representation of the results.

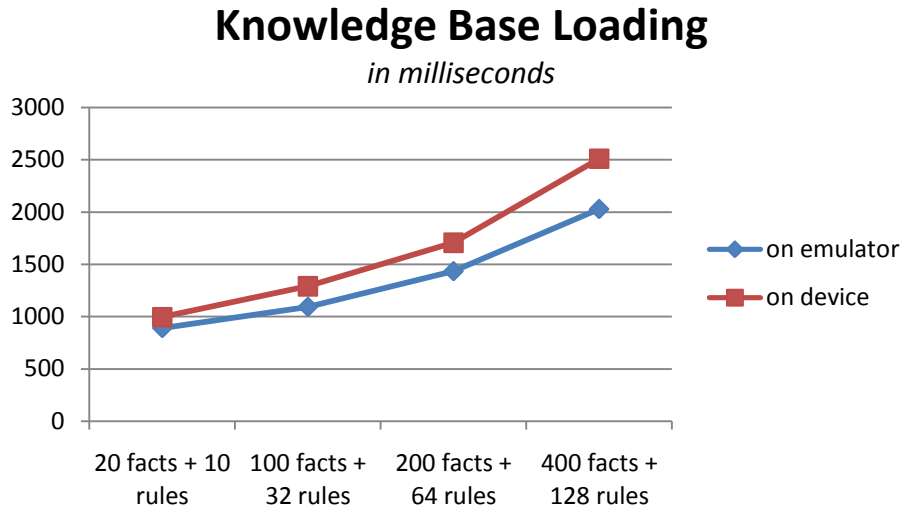



**Chart 6.1:** KB loading times mobile device vs emulator

**Local Query execution.** Here we measured the execution times for queries that can be answered locally based on the previously mentioned (growing) theories, first using a mobile device and then an emulator. For this test, four kinds of queries were used, mainly varying in complexity and of course due to the rule (and fact) structuring of the test theories. The first case is a query returning true, answered deriving immediately from a fact, making this the easiest case. The second query is also proved but using intermediary rules making it more complex to compute. The third kind of query is actually the same with the second but this time the query returns a negative answer because of another conflicting defeasible rule with higher priority. This local conflict resolution is the most complex and time consuming case as is shown by the numbers reported in table 6.4. The last query returns undefined which is

the case when there is no knowledge supporting what we ask or supporting the opposite (negation) of what we ask and is a bit more complex that the first kind due to the algorithms exhausting search for an answer which is not possible since there are missing facts to support each case.

| Kind of Query | 20 facts + 10 rules | 100 facts + 32 rules | 200 facts + 64 rules | 400 facts + 128 rules |
|---|---|---|---|---|
| **ON DEVICE** | | | | |
| **true derived directly from fact** | 3 | 14 | 23 | 45 |
| **true using def rules** | 5 | 168 | 256 | 502 |
| **negative using def rules** | 7 | 367 | 624 | 1287 |
| **undefined** | 21 | 105 | 149 | 312 |
| **ON EMULATOR** | | | | |
| **true derived directly from fact** | 5 | 5 | 16 | 16 |
| **true using def rules** | 31 | 46 | 46 | 62 |
| **negative using def rules** | 78 | 94 | 141 | 209 |
| **undefined** | 19 | 31 | 47 | 47 |

**Table 6.4:** Local Query execution times

Similarly to the theory loading times of the previous described experiment, we can see that the figures for the device and the emulator shown in the table 6.4 are very close when it comes to small knowledge bases (e.g. our scenarios), almost instant responses, but as the KB grows in facts and rules, queries become more demanding. The bigger KB obviously affects the time for query resolution using an emulator but it is much less than the effect it has on the device's restricted capabilities. In complex cases of queries it can take almost 1.3 seconds, 1 second more than the emulator. In easy cases such as of query answers deriving immediately from a fact the device is in the same order of magnitude as the emulator (45ms vs 16 ms), and in easy to typical queries it can take up to half a second for an answer, still much slower than the emulator but acceptable. Summarizing the knowledge base's size as expected greatly affects the evaluation of a query, the number of facts have much less effect (as reflected by the queries that derive immediately from them) than the computational overhead introduced by a complex structured rule set. (e.g. conflicting or chaining) .

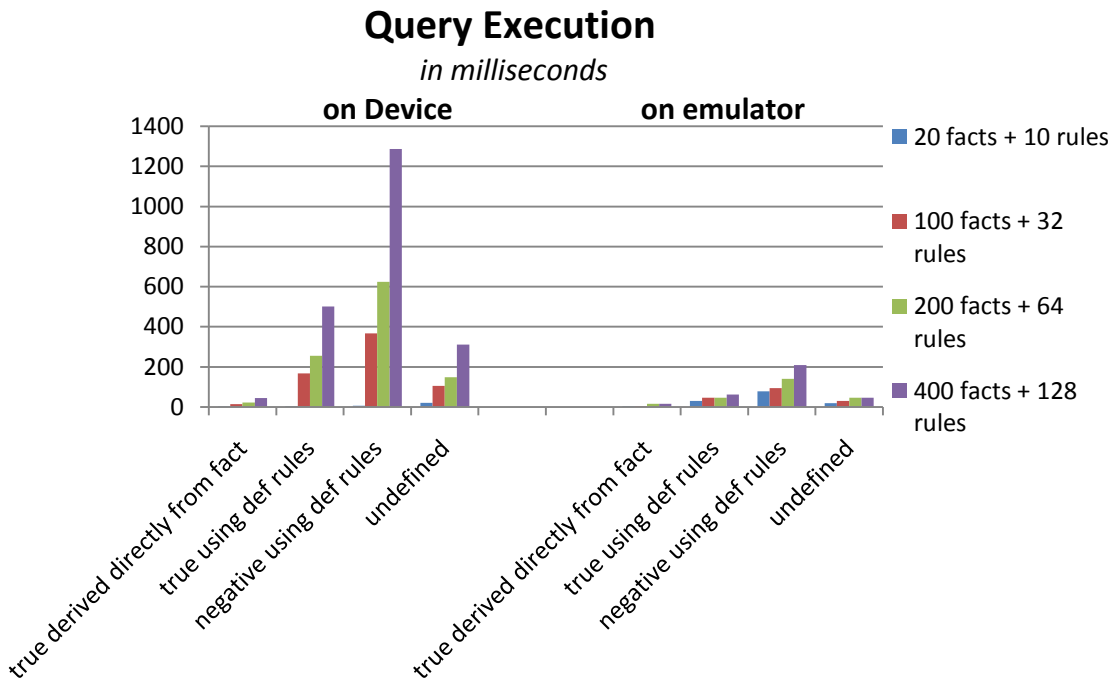



**Chart 6.2:** Local Query execution times, mobile device vs emulator

**Peer-to-Peer query execution: In breadth distribution of knowledge.** For this test we make queries to a mobile device, these queries are each answered by a mapping rule containing a number of foreign literals each of them is acquired through a different source, hence the in breadth knowledge distribution characterization of this experiment. The test was performed first on an emulator then on an actual device, but for both cases three PC's were used to simulate the external sources of information represented in the mappings which were contacted through a real Wi-Fi network for a literal at the time. It has to be noted that the premises of the mapping rules were simple literals therefore the complexity of the query at the external source is the same as a local query derived from a fact using a simple Knowledge Base 10 facts and 10 mapping rules.

Table 6.5 shows timings for peer-to-peer query execution involving mapping rules with different with an increasing number of premises representing different peers. Chart 6.3 depicts the experiment differences in timings between the emulator and the mobile device.

| In breadth Peer-to-Peer query execution | | |
|---|---|---|
| **Number of Peers Involved** | **emulator** | **device** |
| 1 | 37 | 33 |
| 2 | 58 | 62 |
| 4 | 117 | 114 |
| 9 | 269 | 251 |
| 18 | 429 | 486 |
| 36 | 844 | 1015 |
| 54 | 1226 | 1489 |
| 108 | 2232 | 2933 |

**Table 6.5:** In breadth p2p query execution times

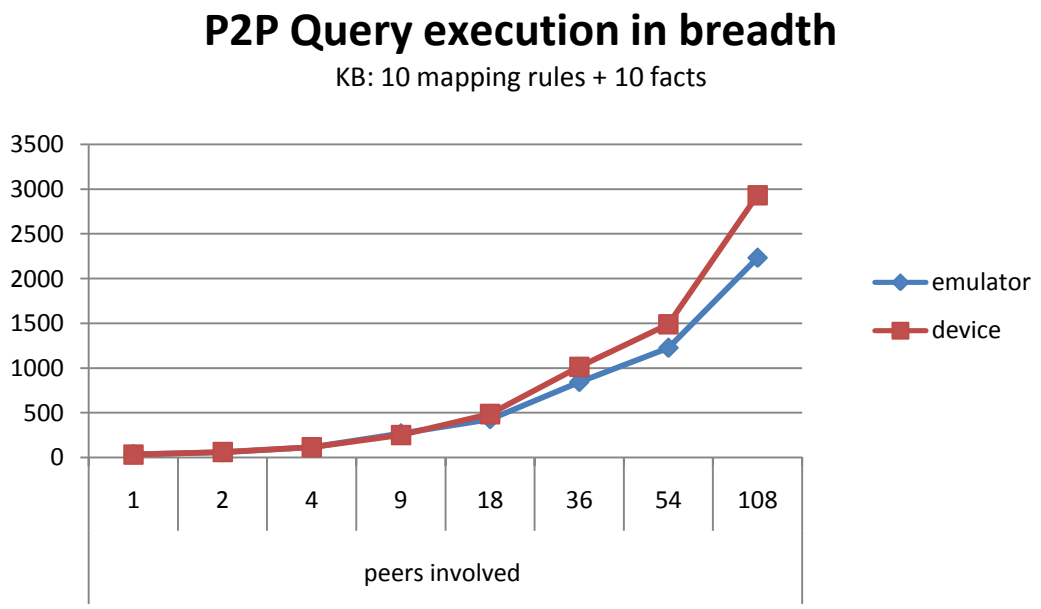



**Chart 6.3:** In breadth p2p query performance, mobile device vs emulator

Moreover, in table 6.6 and chart 6.4 are estimations of the numbers that would occur for queries deriving from facts on bigger Knowledge Bases based on assumptions derived from patterns extracted from observations of the CDL Applications behavior from the local queries experiment described previously.

| | **emulator** | | **device** | |
|---|---|---|---|---|
| **peers involved** | 200 facts + 64 rules | 400 facts + 128 rules | 200 facts + 64 rules | 400 facts + 128 rules |
| 1 | 42 | 44 | 48 | 68 |
| 2 | 68 | 72 | 92 | 132 |
| 4 | 137 | 145 | 174 | 254 |
| 9 | 314 | 332 | 386 | 566 |
| 18 | 519 | 555 | 756 | 1116 |
| 36 | 1024 | 1096 | 1555 | 2275 |
| 54 | 1496 | 1604 | 2299 | 3379 |
| 108 | 2772 | 2988 | 4553 | 6713 |

**Table 6.6:** Estimated values in breadth p2p query performance

More specifically we used the table 6.5 and updated the values, taking into consideration that the query that in the end is evaluated at each involved peer is a local query and therefore by adding the difference in process time occurring from the bigger knowledge base as it occurred on the experiment with local queries and increasing in size knowledge bases.

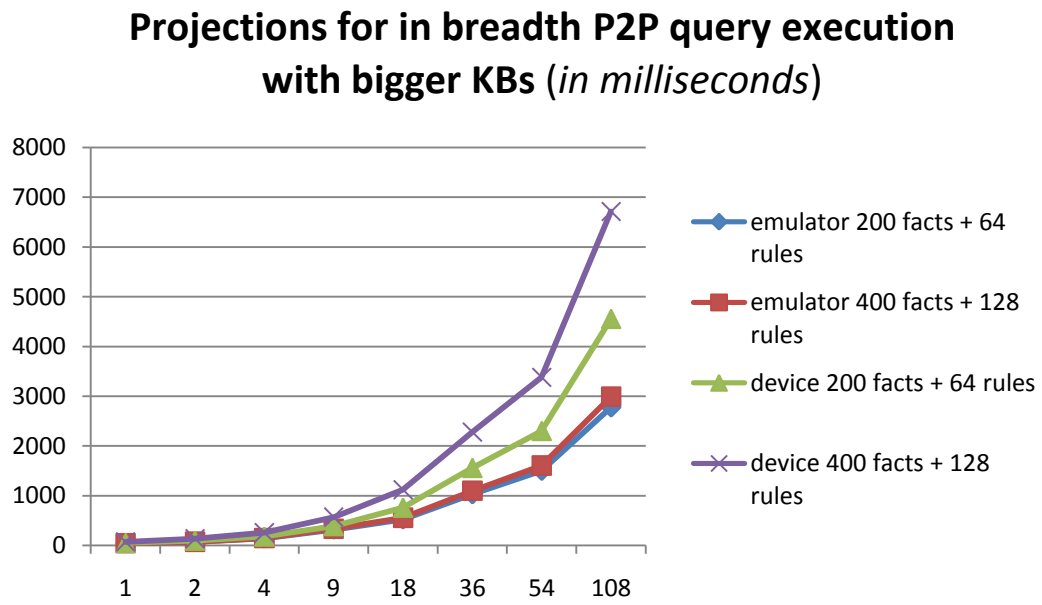



**Chart 6.4:** Estimation of in breadth p2p query performance (projected values)

Following the same paradigm we can project the values for more complex queries and for bigger knowledge bases by using the results of the local queries: e.g. the first record from table 6.5 is the sum of the communication cost (wifi) with another peer and the cost of the query this other peer performs (locally), therefore for this case we add the cost difference of a local query derived from fact and the one from a more complex query e.g. true through using defeasible rules, and the resulting time is the one for a more complex query using the same knowledge base. Chart 6.5 projects an estimation of a more complex query resulting in true and derived from the use of defeasible rules not immediately from a fact. In which we see that the times for the device are escalating very quickly every time the peers involved in an answer increase (double).

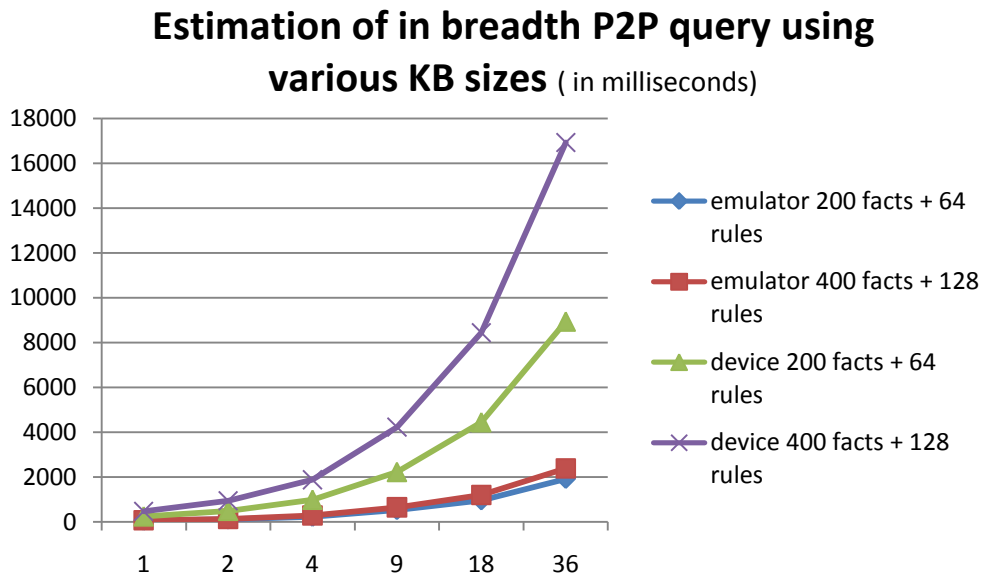



**Chart 6.5:** Estimation of in breadth p2p query performance (complex query)

**Peer-to-Peer query execution: In depth distribution of knowledge.** For this test we make queries to a mobile device, these queries are each answered by a mapping rule containing a foreign literals which is acquired through a different source in which this literal is also answered through a mapping rule and so on…, hence the in depth knowledge distribution characterization of this experiment. The test was performed first on emulators and then on actual devices, but for both cases real Wi-Fi network was used. Similar to the previous experiment, each premise of each mapping rule involved, was a simple literal, therefore the complexity of the query at the end point

mobile device is the same as a local query derived from a fact using a simple Knowledge Base 10 facts and 10 mapping rules. Table 6.7 shows the timings of these experiments for simulation using emulators and actual mobile devices.

| In depth P2P Query execution | | |
|---|---|---|
| **Peers involved (depth)** | **emulator** | **device** |
| 1 | 43 | 47 |
| 2 | 96 | 113 |
| 3 | 139 | 181 |
| 4 | 247 | 361 |
| 5 | 459 | 670 |

**Table 6.7:** In depth p2p query execution times

The experiment depth was kept low due to the complexity of the setup and lack of enough Wi-Fi enabled mobile phones. Chart 6.6 compares graphically the performance of the simulated experiment and the one using real mobile devices (phones).

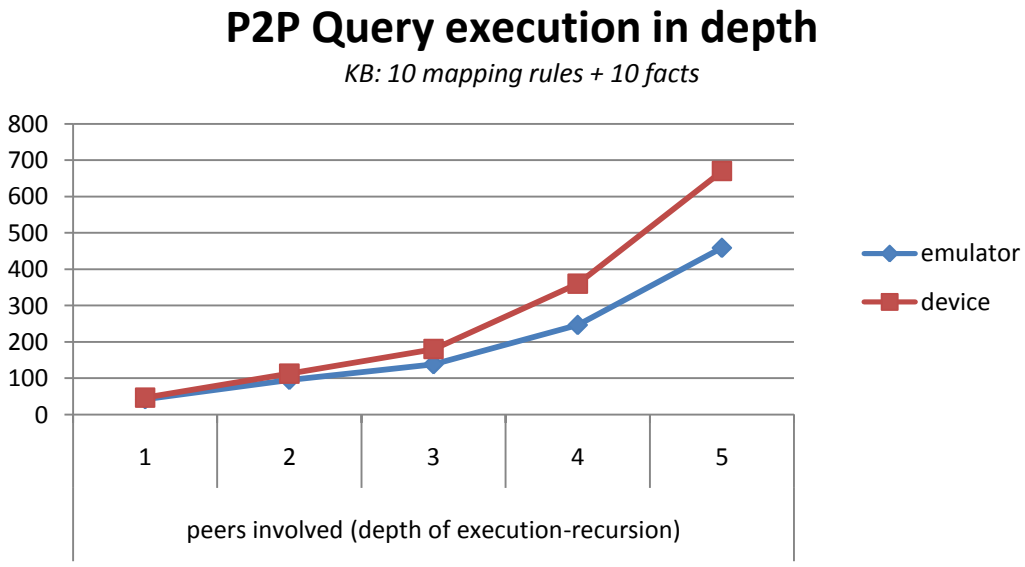



**Chart 6.6:** In depth p2p query performance, emulator vs device

**Peer-to-Peer query execution: Synthetic case.** This experiment uses at first a level distribution on knowledge in breadth, and a second level in depth. Resulting in a tree like execution shown in figure 6.1, in which peer A is the one issuing the query, then

dispatches other queries resulting from the use of a mapping rule, and these other peers also need to use each one more mapping rule to acquire its missing knowledge.

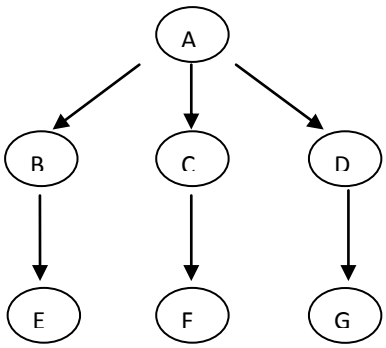



**Figure 6.1:** Synthetic case of in breadth and in depth query execution

Table 6.8 shows the times for the synthetic case of distribution of knowledge shown in figure 6.1, where are available also timings for more complex cases regarding the in depth distribution of knowledge at the first level. The times for the device are estimation based on times occurring from local queries experiments from table 6.4. Chart 6.7 projects the numbers of table 6.8 and depicts the performance of the emulator and an estimation for the device.

| Synthetic case of P2P Query execution | | |
|---|---|---|
| **Peers involved (breadth/depth)** | **emulator** | **device** |
| 1/1 | 46 | 44 |
| 4/4 | 205 | 192 |
| 9/9 | 467 | 389 |
| 18/18 | 825 | 990 |
| 36/36 | 1636 | 2123 |

**Table 6.8:** Synthetic case of p2p query execution

**Experimentation conclusions.** As it becomes obvious from the charts, actual cell phone devices are slower in all aspects (e.g. computation, network communication etc.) compared to testing with cell phone emulators on desktop computers (which actually would even more fast if it did not also emulate the slow hardware of a cell phone). These are used as a measure of comparison for performance for the same tasks. There is room for improvement when the number of nodes or queries increases.

Most probably, as Moore's law suggests their performance will improve as well, to cover such needs. This is the case between the 2 and 3 years old cell phones that we used in these experiments, which had more than 100% difference in computing power, as it was reflected by the time needed for the same tasks rendering the one inadequate and the other sufficient.

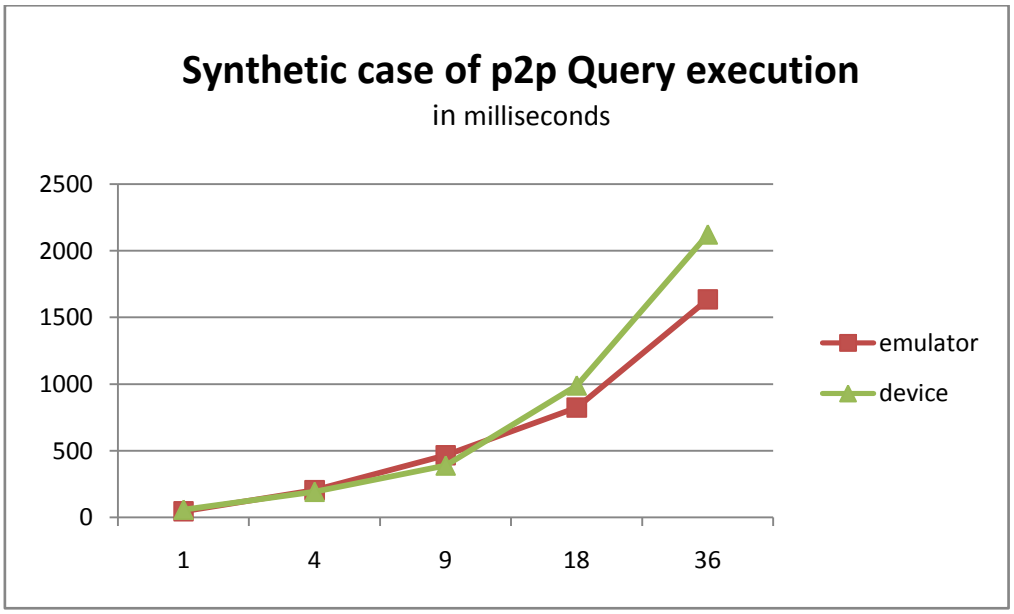



**Chart 6.7:** Synthetic case of p2p query, emulator vs mobile device(estimate)

Recently, based on impressions from the deployment of CDL System on a last generation cell phone (not high end model), the above mentioned expectations for hardware improvements in the future were once again verified. Even though these are impression and not actual time measurements, the cell phone was responding characteristically faster that the older models for similar tasks. This significant improvement keeps going and today there are smart phones running at 1GHz and 512 ram memory as a result this will probably (and it is in our plans for the future to do actual measurements to test this, but after optimizing further the CDL System) permit us to develop and use applications for even more demanding tasks.

# Chapter 7

# Conclusions

To conclude this thesis, we summarize and discuss its main contributions, and propose possible directions for future research.

## 7.1 Synopsis

The imperfect nature of context knowledge and the special characteristics of ambient devices and Ambient Intelligence environments have introduced new challenges in the field of Distributed Artificial Intelligence. Most current Ambient Intelligence systems have not successfully addressed most of them, by relying on unrealistic simplifying assumptions, such as perfect knowledge of context, centralized context, and unbounded computational and communicating capabilities. The requirements, though, are much different in such environments. The uncertainty of context and its distribution to heterogeneous devices with restricted capabilities, impose the need for relaxing these assumptions and for employing different reasoning approaches.

This thesis studies, the problem of reasoning with imperfect context and preference information in Multi-Context Systems. It proposes extensions to the framework described in [20], in which uncertainty in the local theories is modelled using defeasible local theories, while conflicts that may arise from the interaction of contexts through mappings are resolved using contextual preference information. This is demonstrated using a scenario that motivated the current research study. This work also reports on initial experiences gained from the implementation and deployment of contextual defeasible reasoning in real, not simulated ambient intelligence and mobile environments.

In chapter 3 we provide the modified, argumentation-based semantic characterization of the model. To support missing preference information, we use partial preference ordering on the set of contexts. In chapter 4 we present the changes that were brought about to the distributed algorithm for query evaluation that

implements the updated argumentation framework. In chapter 5, the architecture of an implementation on small devices is presented and finally in chapter 6 there is the definition and implementation of two concrete application scenarios and also a discussion of the performance and issues of scalability of the approach. Overall, these findings suggest that simple KR can play an important role in ambient intelligence and pervasive computing: it is rich enough to solve selected problems in these areas, has a formal foundation and semantics and is sufficiently efficient to meet the increased requirements in these environments.

## 7.2 Future Directions

This work is just one step in an ambitious research plan, and there are concrete ideas on further works. So far, our approach assumes that devices/contexts are always willing to disclose information available to them. In future work, we intend to enrich our approach with a mechanism of access control, to address the key issues of privacy and security in AmI environments. In addition, we intend to study other rich forms of KR in ambient intelligence, including agent coordination to solve problems collaboratively, and reasoning about action. Finally, we intend to broaden the scope of contextual reasoning by allowing different types of peers to work together; in particular, we intend to study the use of recent developments in the area of reasoning about context [25].

Future directions of our research are summarized and categorized as follows:

Research in a theoretic level:

- Studying further the complexity of the proposed algorithm, based on the complexity results of our original approach [20]
- Studying alternative methods for conflict resolution, which differ in the way that agents evaluate the imported context information
- Relax some of these assumptions in order to generalize our reasoning methods, and enable their applicability in a greater range of applications.

And in implementation level:

- Extension of the CDL System to support not only small, mobile devices as contexts with full reasoning capabilities but all computers.
- Support the CDL System by creating better interfaces, extending the functionality and optimize its performance.
- Implement real-world applications of our approach in Ambient Intelligence environments, such as those described in [18, 32], but also in other domains with similar requirements such as Social Networks and the Semantic Web.

  The above ambient environments will also include several sensory subsystems, such as Radio Frequency Identification and the Collaborative Location Sensing system [35] which exploits the IEEE 802.11 wireless network infrastructure for positioning, and a multi-camera vision system supporting the development of wide-area entertainment applications [79].

Overall, we believe that ambient intelligence and pervasive computing are a rich testbed for KR: it is a rich area with specific requirements in terms of openness, distribution, heterogeneity and efficiency. Thus it can serve as a source of inspiration and advancement, just as the web has done so in the past decade (semantic web).

# Appendix A

## TuProlog Reasoner

TuProlog [31] is a Java-based light-weight Prolog for Internet applications and infrastructures. For this purpose, tuProlog is designed to feature some interesting qualities: it is easily deployable, just requiring the presence of a Java VM and an invocation upon a single JAR file; its core is both minimal, taking the form of a tiny Java object containing only the most essential properties of a Prolog engine, and configurable, thanks to the loading and unloading of predicates, functors and operators embedded in libraries; the integration between Prolog and Java is as wide, deep, clean as possible; finally, interoperability is developed along the two main lines of Internet standard patterns and coordination models.

## Metaprogram

On the TuProlog reasoning engine we load and run the DR-PROLOG metaprogram. It is a prolog program, implementing the defeasible logic, actually it is a shorter more lightweight version of the original the ideas of which are described in [8] for compatibility with tuProlog engine and taking into consideration the limited resources of mobile devices.

```
supportive_rule(Name, Head, Body) :- strict(Name, Head, Body).

supportive_rule(Name, Head, Body) :- defeasible(Name, Head,
Body).

rule(Name,Head,Body) :- supportive_rule(Name, Head, Body).


definitely(X):- fact(X).

definitely(X):- strict(R,X,L), definitely_provable(L).

definitely(X):- strict0(R,X,L), definitely_provable(L).

definitely(neg(X)):- strict1(R,neg(X),L),
definitely_provable(L), not(definitely(X)).

definitely(X):- strict2(R,X,L), definitely_provable(L).


definitely_provable([]).
```

```
definitely_provable(X):- definitely(X).

definitely_provable([X1|X2]):- definitely_provable(X1),
definitely_provable(X2).


defeasibly(X):- definitely(X).

defeasibly(X):- negation(X,X1), supportive_rule(R,X,L),
defeasibly_provable(L), not(definitely(X1)),
not(overruled(R,X)).

defeasibly_provable([]).

defeasibly_provable(X):- defeasibly(X).

defeasibly_provable([X1|X2]):- defeasibly_provable(X1),
defeasibly_provable(X2).


overruled(R,X):- negation(X,X1), supportive_rule(S,X1,U),
defeasibly_provable(U), not(defeated(S,X1)).

defeated(S,X):- sup(T,S), negation(X,X1),
supportive_rule(T,X1,V), defeasibly_provable(V).


negation(~(X),X):- !.

negation(X,~(X)).


append([],List,List).

append([Head|Tail],List2,[Head|Result]):-
append(Tail,List2,Result).

member(N,[N|Tail]).

member(N,[I|Tail]):- member(N,Tail).


minus_set([E|X],Y,Z):- member(E,Y),minus_set(X,Y,Z),!.

minus_set([E|X],Y,[E|Z]):-minus_set(X,Y,Z),not(member(E,Y)),
!.

minus_set([],Y,[]).

strict(e,w,r).

defeasible(y,t,e).
```

```
fact(w).
sup(e,w).
```

## General CDL System experiment setup

Here we provide the setup of the experiment for the general CDL system testing from section 6.3, more specifically the test theories: profiles and knowledge bases that were used and the queries based on which the occurred our timings we reported.

### Measuring time intervals

For measuring the time needed for certain task we set long integer variables for each interval we were interested in, at the end of each task we used a printing method to report the times recorded. The time is acquired through a java function accessing the system clock, the time it returns is in milliseconds and is the current system time from 1970, therefore we subtract the time recorded immediately after the task we are interested in timing from the time immediately before and this way occurs the measured time. Actually the these times are always a little less than the real time since all the variables and calculations for measuring time and especially the system calls for printing are themselves time consuming and in normal mode would not be used.

### Local Reasoning

Here follow the test data for the tasks of (a) Theory loading and (b) Local queries. To make the knowledge Base bigger for local queries experiment we used dummy data of the form: fact(prefix_data). We created a set of 20 facts and then with *copy- paste* we create more and using the *find-and-replace* search we change the prefix so that they would not conflict. In a similar way we expand the rule base by using a set of 8 rules of the form defeasible( empty rule_name, head ,[premise1, premise2…]). The rule name is empty, the rule head has a prefix a or b or c…and the premises the same prefix respectively. The knowledge base grows by copy pasting these 8 rules and find-and-replace every a with b.

```
//knowledge base dummy data (facts and rules)

knowledgeBase =

//dummy facts

"fact(aa).\n"+"fact(ab).\n"+"fact(ac).\n" + "fact(ad).\n" +
"fact(ae).\n"+"fact(aa).\n"+"fact(ag).\n" + "fact(ah).\n" +
"fact(ai).\n"+"fact(aj).\n"+"fact(ak).\n" + "fact(al).\n" +
"fact(am).\n"+"fact(an).\n"+"fact(ao).\n" + "fact(ap).\n" +
"fact(aq).\n"+"fact(ar).\n"+"fact(as).\n" + "fact(at).\n" +


"fact(ba).\n"+"fact(bb).\n" +"fact(bc).\n" +"fact(bd).\n" +
"fact(be).\n"+"fact(ba).\n" +"fact(bg).\n" +"fact(bh).\n" +
"fact(bi).\n"+"fact(bj).\n" +"fact(bk).\n" +"fact(bl).\n" +
"fact(bm).\n"+"fact(bn).\n" +"fact(bo).\n" +"fact(bp).\n" +
"fact(bq).\n"+"fact(br).\n" +"fact(bs).\n" +"fact(bt).\n" +

//dummy rules

//created by copy past & find-and-replace a with b, b with c …

"defeasible( ,notify_user(event(X,Y,Z)),[interestIn(aw)]).\n"+

"defeasible( ,notify_user(event(X,Y,Z)),[interestIn(az)]).\n"+

"defeasible( ,notify_user(a1), [aa , ak ] ).\n" +

"defeasible( ,notify_user(a2), [ab , al] ).\n" +

"defeasible( ,notify_user(a3), [ac , am ] ).\n" +

"defeasible( ,a4, [ad , an ] ).\n" +

"defeasible( ,a5, [ae , ao ] ).\n" +

"defeasible( ,a6, [af , ap ] ).\n" +


//personal profile preferences (knowledge + defeasible rules)

defeasibleProfile =

"fact(interestIn(mscpres)).\n" +

"fact(busyOn('1-11-2010@13:00-14:00')).\n"+

"defeasible(r1,notify_user(event(X,Y,Z)),[interestIn(X)]).\n"+
```

```
"defeasible(r2, oxi(notify_user(event(X,Y,Z))),[interestIn(X),
busyOn(Y)] ).\n" +

"sup(r2,r1).\n";
```

The queries for this knowledge base are:

- Query resulting in truth deriving immediately from a fact (easy)
  - *defeasibly(interestIn(mscpres))*
- Query resulting in truth deriving from a chaining of defeasible rules (more difficult)
  - *notify_user(event(mscpres,'1-11-2010@13:00-14:00','kwnstantinos papatheodorou master presentation in RA201 at 1-11-2010@13:00-14:00')).*
- Query resulting in no deriving from conflict resolution (even more difficult)
  - *notify_user(event(mscpres,'2-11-2010@13:00-14:00','kwnstantinos papatheodorou master presentation in RA201 at 2-11-2010@13:00-14:00')).*
- Query resulting in undefined, no fact or rule supporting it. (medium difficulty)
  - *defeasibly(nonExistingFact)*

## Distributed reasoning

Here are the dummy data for peer-to-peer querying. The setup of the experiment is as follows: every peer has the same knowledge base. Each mapping rule has a head which is the query we issue and its body contains foreign literals (all are foreign for this experiment) which are acquired through other contexts(peers) by issuing a query to the appropriate peer using the peer addresses provided in the profile using the following form: peerName__foreignLiteral--peerContactAddress , which means that when the algorithm come across a 'foreignLiteral' e.g. in the body of a mapping rule, we use its peerContactAddress to issue a query for it and get its truth value, the name is used for the description of peer preferences. Three different computers were used

on which were spread 9 literals, which were individually asked through the mapping rules, thus simulating up to 108 different peers involved in a query execution.

```
//knowledgeBase

knowledgeBase =

"fact(bb1).\n" + "fact(bb2).\n" + "fact(bb3).\n" +

"fact(bb4).\n" + "fact(bb5).\n" + "fact(bb6).\n" +

"fact(bb7).\n" + "fact(bb8).\n" + "fact(bb9).\n" +

"fact(a1).\n";


//defeasible profile

//mapping rules for in breadth query execution

mappingRules =

"defeasible( rm1, beta1, [ bb1] ).\n" +

"defeasible( rm2, beta2, [ bb1, bb2 ] ).\n" +

"defeasible( rm3, beta4, [ bb1, bb2, bb3, bb4 ] ).\n" +

"defeasible( rm4, beta3, [ bb1, bb2, bb3, bb4, bb5, bb6, bb7,
bb8, bb9 ] ).\n" +

"defeasible( rm5, beta18, [ bb1, bb2, bb3, bb4, bb5, bb6, bb7,
bb8, bb9, bb1, bb2, bb3, bb4, bb5, bb6, bb7, bb8, bb9 ] ).\n";

"defeasible( rm6, beta36, [ bb1, bb2, bb3, bb4, bb5, bb6, bb7,
bb8, bb9, bb1, bb2, bb3, bb4, bb5, bb6, bb7, bb8, bb9 ,bb1,
bb2, bb3, bb4, bb5, bb6, bb7, bb8, bb9, bb1, bb2, bb3, bb4,
bb5, bb6, bb7, bb8, bb9 ] ).\n" +

"defeasible( rm7, beta54, [ bb1,bb2,bb3,bb4, bb5, bb6, bb7,
bb8, bb9, bb1, bb2, bb3, bb4, bb5, bb6, bb7, bb8, bb9, bb1,
bb2, bb3, bb4, bb5, bb6, bb7, bb8, bb9, bb1, bb2, bb3, bb4,
bb5, bb6, bb7, bb8, bb9, bb1, bb2, bb3, bb4, bb5, bb6, bb7,
bb8, bb9, bb1, bb2, bb3, bb4, bb5,bb6,bb7,bb8, bb9 ] ).\n"+

"defeasible( rm8, beta108, [ bb1, bb2, bb3, bb4, bb5, bb6,
bb7, bb8, bb9, bb1, bb2, bb3, bb4, bb5, bb6, bb7, bb8, bb9,
bb1, bb2, bb3, bb4, bb5, bb6, bb7, bb8, bb9, bb1, bb2, bb3,
bb4, bb5, bb6, bb7, bb8, bb9, bb1, bb2, bb3, bb4, bb5, bb6,
```

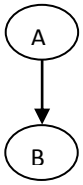
```
bb7, bb8, bb9, bb1, bb2, bb3, bb4, bb5, bb6, bb7, bb8, bb9,
bb1, bb2, bb3, bb4, bb5, bb6, bb7, bb8, bb9, bb1, bb2, bb3,
bb4, bb5, bb6, bb7, bb8, bb9, bb1, bb2, bb3, bb4, bb5, bb6,
bb7, bb8, bb9, bb1, bb2, bb3, bb4, bb5, bb6, bb7, bb8, bb9,
bb1, bb2, bb3, bb4, bb5, bb6, bb7, bb8, bb9, bb1, bb2, bb3,
bb4, bb5, bb6, bb7, bb8, bb9 ] ).\n" +


// mapping rules for in depth query execution

"defeasible( rm9, alpha, [ beta] ).\n" +

"defeasible( rm10, beta, [ gamma ] ).\n" +

"defeasible( rm11, gamma, [ delta ] ).\n" +

"defeasible( rm12, delta, [ epsilon ] ).\n" +

"defeasible( rm13, epsilon, [ bb7 ] ).\n" +

"defeasible( rm14, zita, [ bb8 ] ).\n" +

"defeasible( rm15, hta, [ bb9 ] ).\n" +


// mapping rules for synthetic cases of in breadth and in
depth query

“defeasible( rm16, synthetic3, [ epsilon, zita, hta ]).\n”+

“defeasible( rm17, synthetic6, [ epsilon, zita, hta, epsilon,
zita, hta ]).\n” +

"defeasible( rm18, synthetic9, [epsilon, zita, hta, epsilon,
zita, hta, epsilon, zita, hta] ).\n";


//peer information of the form:

//peerName__foreignLiteral--peer address

peerInfo =

"p1__bb1--socket://139.91.183.35:5491\n" +

"p2__bb2--socket://139.91.183.35:5491\n" +

"p3__bb3--socket://139.91.183.35:5491\n" +
```

```
"p4__bb4--socket://139.91.183.30:5491\n" +
"p5__bb5--socket://139.91.183.30:5491\n" +
"p6__bb6--socket://139.91.183.30:5491\n" +
"p7__bb7--socket://139.91.183.47:5491\n" +
"p8__bb8--socket://139.91.183.47:5491\n" +
"p9__bb9--socket://139.91.183.47:5491\n" +

"p1__beta1--socket://139.91.183.35:5491\n" +
"p2__beta2--socket://139.91.183.35:5491\n" +
"p3__beta4--socket://139.91.183.35:5491\n" +
"p4__beta9--socket://139.91.183.30:5491\n" +
"p5__beta18--socket://139.91.183.30:5491\n" +
"p6__beta36--socket://139.91.183.30:5491\n" +
"p7__beta54--socket://139.91.183.47:5491\n" +
"p8__beta108--socket://139.91.183.47:5491\n" +

"p1__alpha--socket://139.91.183.35:5491\n" +
"p2__beta--socket://139.91.183.35:5491\n" +
"p3__gamma--socket://139.91.183.35:5491\n" +
"p4__delta--socket://139.91.183.30:5491\n" +
"p7__epsilon--socket://139.91.183.47:5491\n" +
"p8__zita--socket://139.91.183.47:5491\n" +
"p9__hta--socket://139.91.183.47:5491\n" +
"p1__synthetic3--socket://139.91.183.35:5491\n" +
"p2__synthetic6--socket://139.91.183.35:5491\n" +
"p3__synthetic9--socket://139.91.183.35:5491\n" +
```

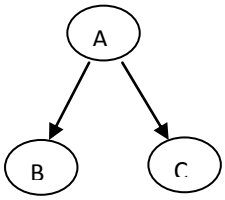
```
// preference – trust over other MCS peers
peerPreference =   "p2__p1__p3__p4__p5__p6__p7__p8__p9";
```

**The distributed (p2p) queries for this Multi-Context System are:**

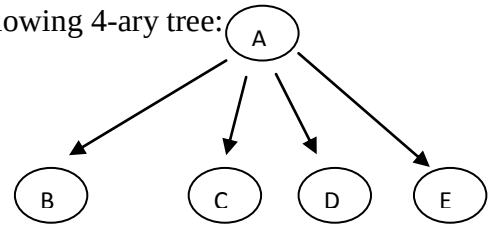
- **In breadth distribution of knowledge**
  - *defeasibly(beta1)*
    - containing 1 foreign literal to be acquired from another system peer by issuing a Wi-Fi query for this literal resulting in the following graphical representation of the query execution:



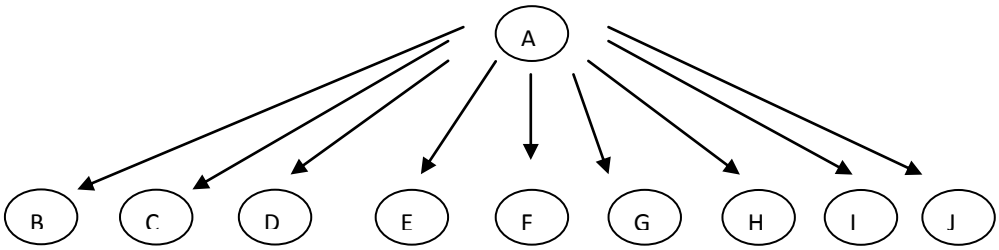

  - *defeasibly(beta2)*
    - containing 2 foreign literals to be acquired from other system peers by issuing individual Wi-Fi queries for each literal, resulting in the following binary execution tree:




  - *defeasibly(beta4)*
    - containing 4 foreign literals to be acquired from other system peers by issuing individual Wi-Fi queries for each literal resulting in the following 4-ary tree:



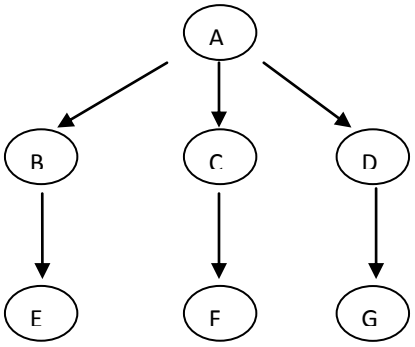

  - *defeasibly(beta9)*
    - containing 9 foreign literals to be acquired from other system peers by issuing individual Wi-Fi queries for each literal resulting in a 9-ary tree with height 1.



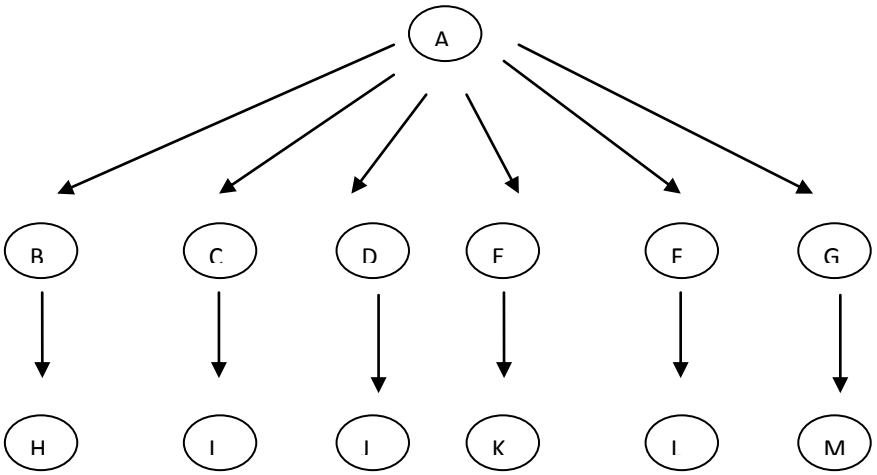

- *defeasibly(beta18)*
  - similarly to the above cases
- *defeasibly(beta36)*
- *defeasibly(beta54)*
- *defeasibly(beta108)*

- **In depth distribution of knowledge**
  - *defeasibly(alpha)*
    - triggering a recusion of level 5 for answering this query
  - *defeasibly(beta)*
    - triggering a recursion of level 4 for answering this query
  - *defeasibly(gamma)*
    - triggering a recursion of level 3 for answering this query
  - *defeasibly(delta)*
    - triggering a recursion of level 2 for answering this query
  - *defeasibly(epsilon)*
    - triggering a recursion of level 1 for answering this query

- **Synthetic case of distribution of knowledge**
  - *defeasibly(synthetic3)*
    - the peer on which the query is issued, finds that the premises of the mapping rule this query fits to has 3 foreign literals, issues these queries and each of these peers also issues (recursion) one query to another peer. Therefore for this execution of this query a tree of the following form occurs:




- *defeasibly(synthetic6)*
  - the peer on which the query is issued, finds that the premises of the mapping rule this query fits to has 3 foreign literals, issues these queries and each of these peers also issues (recursion) one query to another peer. Therefore for this execution of this query a tree of the following form occures:




- *defeasibly(synthetic9)*
  - the peer on which the query is issued, finds that the premises of the mapping rule this query fits to has 9 foreign literals, issues these queries and each of these peers also issues (recursion) one query to another peer. Therefore for the execution of this query a 9-ary with height 2 occurs.

The execution times for queries of each case (in breadth, in depth and synthetic), are actually the sum of the time needed for all local queries at the endpoints and the network communication time.

## Related Publications

1) Grigoris Antoniou, Constantinos Papatheodorou, Antonis Bikakis. Reasoning about Context in Ambient Intelligence Environments: A Report from the Field. In Proc. 12th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2010), accepted.

2) Constantinos Papatheodorou, Grigoris Antoniou, Antonis Bikakis. On the Deployment of Contextual Reasoning in Ambient Intelligence Environments. In Proc. 6th International Conference on Intelligent Environments (IE'10), accepted.

3) Grigoris Antoniou, Antonis Bikakis, Constantinos Papatheodorou. Reasoning with Imperfect Context and Preference Information in Multi-Context Systems. In Proc. ADBIS 2010, LNCS XXXX, Springer 2010, invited.

## Bibliography

[1] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a Better Understanding of Context and Context-Awareness. In HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, pages 304:307. Springer-Verlag, 1999. 1, 2

[2]. Philippe Adjiman, Philippe Chatalic, Francois Goasdoue', Marie-Christine Rousset, and Laurent Simon. Distributed Reasoning in a Peer-to-Peer Setting: Application to the Semantic Web. Journal of Artificial Intelligence Research, 25:269:314, 2006. 24, 32, 66

[3] Leila Amgoud and Claudette Cayrol. On the Acceptability of Arguments in Preference-based Argumentation. In UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pages 1:7, 1998. 31, 105

[4] Leila Amgoud and Claudette Cayrol. A Reasoning Model Based on the Production of Acceptable Arguments. Annals of Mathematic and Artificial Intelligence, 34(1-3):197:215, 2002. 31, 105

[5] Leila Amgoud, Claudette Cayrol, and Daniel Le Berre. Comparing Arguments using Preference Orderings for Argument-based Reasoning. 141 In IEEE

International Conference on Tools with Artificial Intelligence ICTAI'96, pages 400:403, Los Alamitos, California, 1996. IEEE Computer Society Press. 31, 105

[6] Leila Amgoud, Simon Parsons, and Laurent Perrussel. An Argumentation Framework based on contextual Preferences. In International Conference on Formal and Applied and Practical Reasoning (FAPR'00), pages 59:67, 2000. 31, 32, 105

[7] Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher. Representation results for defeasible logic. ACM Transactions on Computational Logic, 2(2):255:287, 2001. 29, 34, 49, 66, 71, 102, 104, 137, 138

[8] Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher. Embedding defeasible logic into logic programming. Theory Pract. Log. Program., 6(6):703:735, 2006. 94, 104

[9] Grigoris Antoniou, David Billington, Guido Governatori, and Michale J. Maher. A Flexible Framework for Defeasible Logics. In Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pages 405:410. AAAI Press / The MIT Press, 2000. 29

[10] Grigoris Antoniou and Frank vanHarmelen. A Semantic Web Primer. MIT Press, Cambridge, MA, USA, 2004. 20

[11] G. Attardi and M. Simi. A formalization of viewpoints. Fundamenta Informaticae, 23:149:173, 1995. 21

[12] T.J.M. Bench-Capon. Persuasion in Practical Argument Using Valuebased Argumentation Frameworks. Journal of Logic and Computation, 13:429:448, June 2003. 31, 105 142

[13] M. Benerecetti, F. Giunchiglia, and L. Serafini. Model checking multiagent systems. Journal of Logic and Computation, 8:8:3, 1998. 20

[14] Massimo Benerecetti, Paolo Bouquet, and Chiara Ghidini. Formalizing Belief Reports - The Approach and a Case Study. In AIMSA, pages 62:75, 1998. 21

[15] Massimo Benerecetti, Paolo Bouquet, and Chiara Ghidini. Contextual reasoning distilled. JETAI, 12(3):279:305, 2000. 19

[16] Antonis Bikakis. Defeasible Contextual Reasoning in Ambient Intelligence. PhD Thesis, *Computer Science Department, University of Crete, 2009.*

[17] Antonis Bikakis and Grigoris Antoniou. Defeasible Contextual Reasoning with Arguments in Ambient Intelligence. IEEE Transactions on Knowledge and Data Engineering, accepted. 17

[18] Antonis Bikakis and Grigoris Antoniou. Distributed Defeasible Contextual Reasoning in Ambient Computing. In Proceedings of European Conference on Ambient Intelligence (AmI 08), 5355 of Lecture Notes in Computer Science, pages 308:325. Springer, 2008. 16 143

[19] Antonis Bikakis and Grigoris Antoniou. Local and Distributed Defeasible Reasoning in Multi-Context Systems. In Proceedings of International Symposium on Rule Representation, Interchange and Reasoning on the Web (RuleML 2008), 5321 of Lecture Notes in Computer Science, pages 135:149. Springer, 2008. 17

[20] Antonis Bikakis and Grigoris Antoniou. Contextual Argumentation in Ambient Intelligence. In 10th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09), 2009. accepted. 16

[21] Arnold Binas and Sheila A. Peer-to-Peer Query Answering with Inconsistent Knowledge. In KR, pages 329:339, 2008. 32

[22] Arnold Binas and Sheila A. McIlraith. Exploiting Preferences over Information Sources to Efficiently Resolve Inconsistencies in Peer-topeer Query Answering. In AAAI 2007 Workshop on Preference Handling for Artificial Intelligence, 2007. 24, 25, 32 144

[23] Andrei Bondarenko, Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. An Abstract, Argumentation-Theoretic Approach to Default Reasoning. Artificial Intelligence, 93:63:101, 1997. 26

[24] Paolo Bouquet and Fausto Giunchiglia. Reasoning about Theory Formulation and Reformulation: A New Solution to the Qualification Problem. In Proceedings of the Second World Conference on the Fundamentals of Artificial Intelligence, pages 39:50, 1995. 21

[25] G. Brewka, T. Eiter: Argumentation Context Systems: A Framework for Abstract Group Argumentation. In Proc. LPNMR 2009: 44-57.

[26] Gerhard Brewka, Floris Roelofsen, and Luciano Serafini. Contextual Default Reasoning. In IJCAI, pages 268:273, 2007. 23, 105

[27] Sasa Buvac and Ian A. Mason. Propositional Logic of Context. In AAAI, pages 412:419, 1993. 19

[28] Philippe Chatalic, Gia Hien Nguyen, and Marie-Christine Rousset. Reasoning with Inconsistencies in Propositional Peer-to-Peer Inference Systems. In ECAI, pages 352:356, 2006. 24, 25, 32 145

[29] Harry Chen, Tim Finin, and Anupam Joshi. SemanticWeb in a Pervasive Context-Aware Architecture. Artificial Intelligence in Mobile System 2003, pages 33:40, October 2003. 5, 8

[30] Alessandro Cimatti. Multi-agent reasoning with belief contexts: the approach and a case study. In ECAI-94: Proceedings of the workshop on agent theories, architectures, and languages on Intelligent agents, pages 71:85, New York, NY, USA, 1995. Springer-Verlag New York, Inc. 20

[31] Enrico Denti, Andrea Omicini, Alessandro Ricci, tuProlog: A Light-Weight Prolog for Internet Applications and Infrastructures, Practical Aspects of Declarative Languages, 3rd International Symposium (PADL 2001), Las Vegas, NV, USA, 11-12 March 2001. Proceedings. LNCS 1990, Springer-Verlag, 2001.

[32] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence, 77:321:357, 1995. 26, 36, 102

[33] Michael Fisher and Chiara Ghidini. Programming Resource-Bounded Deliberative Agents. In IJCAI, pages 200:205, 1999. 21

[34] J. Forstadius, O. Lassila, and T. Seppanen. RDF-based model for context-aware reasoning in rich service environment. In PerCom 2005 Workshops, pages 15:19, 2005. 5, 8

[35] Charalampos Fretzagias and Maria Papadopouli. Cooperative Location Sensing for Wireless Networks. In Second IEEE International conference on Pervasive Computing and Communications, Orlando, Florida, 2004. 106

[36] Fabien L. Gandon and Norman M. Sadeh. Semantic web technologies to reconcile privacy and context awareness. Journal of Web Semantics, 1:241:260, 2004. 5, 8 146

[37] Alejandro Javier Garcia and Guillermo Ricardo Simari. Defeasible Logic Programming: An Argumentative Approach. Theory and Practice of Logic Programming, 4(1-2):95:138, 2004. 30

[38] Chiara Ghidini. Modelling (Un)Bounded Beliefs. In CONTEXT '99: Proceedings of the Second International and Interdisciplinary Conference on Modeling and Using Context, pages 145:158, London, UK, 1999. Springer-Verlag. 21

[39] Chiara Ghidini and Fausto Giunchiglia. Local Models Semantics, or contextual reasoning=locality+compatibility. Artificial Intelligence, 127(2):221:259, 2001. 19

[40] Fausto Giunchiglia. Non-Omniscient Belief as Context-Based Reasoning. In Proc. of the 13th International Joint Conference on Artificial Intelligence, pages 548:554, 1993. 21, 22

[41] Fausto Giunchiglia and Enrico Giunchiglia. Ideal and real belief about belief: some intuitions. In MAAMAW '96: Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : agents breaking away, pages 1:12, Secaucus, NJ, USA, 1996. Springer-Verlag New York, Inc. 21

[42] Fausto Giunchiglia and Luciano Serafini. Multilanguage hierarchical logics, or: how we can do without modal logics. Artificial Intelligence, 65(1), 1994. 19, 101

[43] Li Gong. JXTA: A Network Programming Environment. IEEE Internet Computing, 5(3):88:95, 2001. 88

[44] Georg Gottlob. Complexity Results for Nonmonotonic Logics. Journal of Logic and Computation, 2(3):397:425, 1992. 23

[45] Guido Governatori and Michael J. Maher. An Argumentation- Theoretic Characterization of Defeasible Logic. In ECAI, pages 469:473, 2000. 29, 30, 31 147

[46] Guido Governatori, Michael J. Maher, David Billington, and Grigoris Antoniou. Argumentation Semantics for Defeasible Logics. Journal of Logic and Computation, 14(5):675:702, 2004. 29, 32, 36, 102, 105, 139

[47] Tao Gu, Hung Keng Pung, and Da Qing Zhang. A Middleware for Building Context-Aware Mobile Services. In Proceedings of the IEEE Vehicular Technology Conference (VTC 2004), Milan, Italy, May 2004. 5, 8

[48] M. Hatala, R. Wakkary, and L. Kalantari. Ontologies and Rules in Support of Real-time Ubiquitous Application. Journal of Web Semantics, Special Issue on "Rules and ontologies for Semantic Web", 3(1):5:22, 2005. 5, 8

[49] Karen Henricksen and Jadwiga Indulska. Modelling and Using Imperfect Context Information. In Proceedings of PERCOMW '04, pages 33:37, Washington, DC, USA, 2004. IEEE Computer Society. 2

[50] Souhila Kaci and Leendert van der Torre. Preference-based argumentation: Arguments supporting multiple values. Internation Journal of Approximate Reasoning, 48(3):730:751, 2008. 31, 105

[51] Antonis C. Kakas and Pavlos Moraitis. Argumentation based decision making for autonomous agents. In AAMAS, pages 883:890, 2003. 30, 31

[52] Deepali Khushraj, Ora Lassila, and Tim Finin. sTuples: Semantic Tuple Spaces. In First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous.04), pages 267:277, August 2004. 6, 8

[53] Anders Kofod-Petersen and Marius Mikalsen. Representing and Reasoning about Context in a Mobile Environment. Revue d'Intelligence Artificielle, 19(3):479:498, 2005. 5, 8

[54] Panu Korpipaa, Jani Mantyjarvi, Juha Kela, Heikki Keranen, and Esko-Juhani Malm. Managing Context Information in Mobile Devices. IEEE Pervasive Computing, 02(3):42:51, 2003. 6, 8

[55] Reto Krummenacher, Jacek Kopecký, and Thomas Strang. Sharing Context Information in Semantic Spaces. In OTM Workshops, pages 229: 232, 2005. 6, 8

[56] John McCarthy. Generality in Artificial Intelligence. Communications of the ACM, 30(12):1030:1035, 1987. 19

[57] John McCarthy and Saša Buvač. Formalizing Context (Expanded Notes). In Atocha Aliseda, Rob van Glabbeek, and Dag Westerstfahl, editors, Computing Natural Language, pages 13:50. CSLI Publications, Stanford, California, 1998. 19

[58] Sanjay Modgil. Reasoning about preferences in argumentation frameworks. Artificial Intelligence, 2009. 32

[59] S. Parsons, C. Sierra, and N. R. Jennings. Agents that reason and negotiate by arguing. Journal of Logic and Computation, 8(3):261:292, 1998. 20, 21

[60] John L. Pollock. Defeasible Reasoning. Cognitive Science, 11(4):481:518, 1987. 26, 27

[61] Henry Prakken and Giovanni Sartor. Argument-Based Extended Logic Programming with Defeasible Priorities. Journal of Applied Non- Classical Logics, 7(1), 1997. 29, 30, 31

[62] Anand Ranganathan and Roy H. Campbell. An infrastructure for context-awareness based on first order logic. Personal Ubiquitous Comput., 7(6):353:364, 2003. 5, 8

[63] Floris Roelofsen and Luciano Serafini. Minimal and Absent Information in Contexts. In IJCAI, pages 558:563, 2005. 23

[64] Ronald Prescott Loui. Defeat among arguments: a system of defeasible inference. Computational Intelligence, 3:100:106, 1987. 26

[65] Jordi Sabater, Carles Sierra, Simon Parsons, and Nicholas R. Jennings. Engineering Executable Agents using Multi-context Systems. Journal of Logic and Computation, 12(3):413:442, 2002. 20

[66] Bill Schilit and M. Theimer. Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5):22:32, 1994. 1

[67] Luciano Serafini and Paolo Bouquet. Comparing formal theories of context in AI. Artificial Intelligence, 155(1-2):41:67, 2004. 19

[68] Guillermo Ricardo Simari and Ronald Prescott Loui. A Mathematical Treatment of Defeasible Reasoning and its Implementation. Artificial Intelligence, 53(2-3):125:157, 1992. 30

[69] C. Stephanidis, A. A. Argyros, D. Grammenos, X. Zabulis: Pervasive Computing @ ICS-FORTH. Workshop Pervasive Computing @ Home, International Conference on Pervasive Computing 2008.

[70] Jonathan Stillman. The Complexity of Propositional Default Logics. In AAAI, pages 794:799, 1992. 23

[71] Frieder Stolzenburg, Alejandro Javier Garcia, Carlos Ivan Chesnevar, and Guillermo Ricardo Simari. Computing Generalized Specificity. Journal of Applied Non-Classical Logics, 13(1):87:113, 2003. 30

[72] Matthias Thimm. In Proceedings of the Sixth International Workshop on Argumentation in Multi-Agent Systems (ArgMAS'09), pages 155:172, May. 30

[73] Matthias Thimm, Alejandro J. Garca, Gabriele Kern-Isberner, and Guillermo R. Simari. Using Collaborations for Distributed Argumentation with Defeasible Logic

Programming. In M. Pagnucco and 152 M. Thielscher, editors, Proceedings of the 12th International Workshop on Non- Monotonic Reasoning (NMR'08), pages 179:188. University of New South Wales, September 2008. 30

[74] Matthias Thimm and Gabriele Kern-Isberner. A Distributed Argumentation Framework using Defeasible Logic Programming. In Philippe Besnard, Sylvie Doutre, and Anthony Hunter, editors, Proceedings of the 2nd International Conference on Computational Models of Argument (COMMA'08), number 172 in Frontiers in Artificial Intelligence and Applications, pages 381:392. IOS Press, May 2008. 30

[75] Tim Berners-Lee and James Hendler and Ora Lassila. The Semantic Web. Scientific American, May 2001. 20

[76] Alessandra Toninelli, Rebecca Montanari, Lalana Kagal, and Ora Lassila. A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments. In Proc. of 5th International Semantic Web Conference, pages 5:9, November 2006. 4, 5, 8

[77] Xiao Hang Wang, J. S. Dong, C. Y. Chin, S. R. Hettiarachchi, and D. Zhang. Semantic Space: an infrastructure for smart spaces. IEEE Pervasive Computing, 3(3):32:39, 2004. 4, 8 153

[78] Mark Weiser. The Computer for the 21st Century. Scientific American, 265(3):94:104, September 1991. 1

[79] Xenophon Zabulis, Thomas Sarmis, Dimitris Grammenos, and Antonis Argyros. A multicamera vision system supporting the development of wide-area exertainment applications. In IAPR Conference on Machine Vision and Applications (MVA'09), 2009. 106