Under consideration for publication in Knowledge and Information
Systems

Strategies for Contextual Reasoning with Conflicts in Ambient Intelligence

Antonis Bikakis[1], Grigoris Antoniou[1] and Panayiotis Hassapis[2]

[1]Institute of Computer Science, FO.R.T.H., Greece;
[2] Informatics Department of the Athens University of Economics and Business, Greece

Abstract. Ambient Intelligence environments host various agents that collect, process, change and share the available context information. The imperfect nature of context, the open and dynamic nature of such environments and the special characteristics of ambient agents have introduced new research challenges in the study of Distributed Artificial Intelligence. This paper proposes a solution based on the Multi-Context Systems paradigm, according to which local knowledge of ambient agents is encoded in rule theories (*contexts*), and information flow between agents is achieved through mapping rules that associate concepts used by different contexts. To resolve potential inconsistencies that may arise from the interaction of contexts through their mappings (*global conflicts*), we use a preference ordering on the system contexts, which may express the confidence that an agent has in the knowledge imported by other agents. On top of this model, we have developed four alternative strategies for *global conflicts* resolution, which mainly differ in the type and extent of context and preference information that is used to resolve potential conflicts. The four strategies have been respectively implemented in four versions of a distributed algorithm for query evaluation and evaluated in a simulated P2P system.

Keywords: Distributed Reasoning; Contextual Reasoning; Reasoning with Preferences; Ambient Intelligence

1. Introduction

Ambient Intelligence systems aim at providing the right information to the right users, at the right time, in the right place, and on the right device. In order to achieve this, a system must have a thorough knowledge and, as one may say, "*understanding*" of its environment, the people and devices that exist in it, their

Received xxx
Revised xxx
Accepted xxx

interests and capabilities, and the tasks and activities that are being undertaken. All this information falls under the notions of *context*.

The study of Ambient Intelligence environments and pervasive computing systems has introduced new research challenges in the field of Distributed Artificial Intelligence. These are mainly caused by the imperfect nature of context and the special characteristics of such environments and the agents that operate in them. Henricksen and Indulska in (2004) characterize four types of imperfect context information: *unknown*, *ambiguous*, *imprecise*, and *erroneous*. Sensor or connectivity failures, which are inevitable in wireless connections, result in situations that not all context data is available at any time. When data about a context property comes from multiple sources, then context may become ambiguous. Imprecision is common in sensor-derived information, while erroneous context arises as a result of human or hardware errors.

Ambient agents are expected to have different goals, experiences and perceptive capabilities. They may use distinct vocabularies to describe their context; they may even have different levels of sociality. As in all types of multi-agent systems, agents belonging to a broader multi-agent population need to take into account other agents' beliefs while realizing their behavior (Nguyen and Katarzyniak, 2009). However, in this case, due to the highly dynamic and open nature of ambient environments (various entities may join and leave at random times) and the unreliable wireless communications that are restricted by the range of the transmitters, ambient agents do not typically know a priori all other entities that are present at a specific time instance nor can they communicate directly with all of them.

So far, Ambient Intelligence systems have not managed to efficiently handle all these challenges. As it has been already surveyed in (Bikakis et al., 2008), most of them follow classical reasoning approaches that assume perfect knowledge of context, failing to deal with cases of missing, inaccurate or inconsistent context information. Regarding the distribution of reasoning tasks, a common approach followed in most systems assumes the existence of a central entity, which is responsible for collecting and reasoning with all available context information. However, Ambient Intelligence environments have much more demanding requirements. The dynamics of the network and the unreliable and restricted wireless communications inevitably call for decentralization of the reasoning tasks.

In this paper, we propose a fully distributed approach for reasoning in Ambient Intelligence environments, which is based on the Multi-Context Systems paradigm (Giunchiglia and Serafini, 1994; Ghidini and Giunchiglia, 2001). Our approach models ambient agents as autonomous logic-based entities, the knowledge possessed by an agent as a local context rule theory, and associations between the knowledge possessed by different ambient agents as mappings between their respective context theories. To handle uncertainty in the local context knowledge, context theories are modeled as theories of Defeasible Logic (Antoniou et al., 2001). But, even if it is assumed that each context theory is locally consistent, the same assumption will not necessarily hold for the global knowledge base. The union of local theories may result in inconsistencies caused by the mappings. For example, a local theory A may import conflicting information from two different agents, B and C, through two competing mapping rules. In this case, even if the three different theories are locally consistent, their union through the mappings defined by A may contain inconsistencies. To deal with this type of inconsistencies (*global conflicts*), we model mappings as defeasible rules, and use context and preference information to resolve this type of conflicts.

The contribution of this paper is threefold. First of all, it aims at highlighting the challenges of contextual reasoning in Ambient Intelligence environments. This is achieved through three representative scenarios from the Ambient Intelligence domain. Secondly, it proposes a context representation model, which is based on the Multi-Context Systems paradigm, and describes four different versions of a distributed algorithm for query evaluation. Each version implements a different strategy for global conflicts resolution with respect to the type and extent of context and preference information used to resolve the potential conflicts. Finally, it compares the four strategies based on their formal properties, as well as on the results of an experimental evaluation in a simulated P2P system.

The rest of the paper is structured as follows. The next section describes three motivating scenarios from the Ambient Intelligence domain and discusses their special characteristics. Section 3 presents background information and related work on Ambient Intelligence systems, Multi-Context Systems and Peer Data Management Systems. Section 4 presents the proposed context representational model, while Section 5 describes our reasoning approach in the form of a distributed algorithm for query evaluation and studies its formal properties. Section 6 describes three alternative strategies for conflict resolution. Section 7 presents a prototypical implementation of the four strategies in a simulated peer-to-peer system and the results of an experimental evaluation. Section 8 discusses issues related to the selection of the appropriate strategy by an ambient agent and the combination of different strategies in a Multi-Context System. The last section summarizes and presents the next steps of this work.

2. Motivating Scenarios from the Ambient Intelligence Domain

Below, we describe three use case scenarios from the Ambient Intelligence domain. The aim of these scenarios is to highlight the special challenging characteristics of contextual reasoning in Ambient Intelligence.

2.1. Context-Aware Mobile Phone in an Ambient Classroom

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 calls) based on his preferences and context. Dr. Amber has the following preferences: His phone should ring in case of an incoming call, unless it is in silent mode or he is giving a lecture.

Consider the case that Dr. Amber is currently located in the 'RA201' university 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 call, while it is in normal mode.

The phone cannot decide whether it should ring based only on its local context knowledge, which includes information about incoming calls and the mode of the phone, as it is not aware of other important context parameters (e.g. 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.

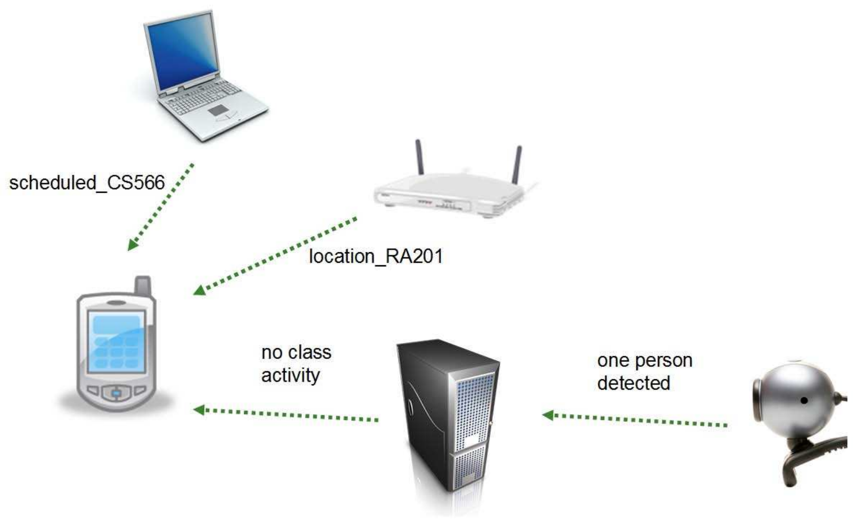

Fig. 1. Context Information Flow in the Scenario

In order 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 Dr. Amber's calendar. According to this, there is a scheduled class event at the time of the call. Information about Dr. Amber's current position is imported from the wireless network localization service, which at the time of the call 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 local information about the state of the projector (it is off), and information about the presence of people in the classroom that it imports from an external person detection service, which in this case detects only one person, the classroom manager infers that there is no class activity.

The overall information flow in the scenario is depicted in Figure 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.

2.2. Ambient Intelligence Home Care System

The second scenario takes place in an apartment hosting an old man, Mr. Jones. Mr. Jones, a 80 year old widower, is living alone in his apartment, while his son resides in close proximity. A nurse visits Mr. Jones 8 to 10 hours daily, while his son also visits him for some hours every couple of days. Mr Jones' apartment is equipped with an Ambient Intelligence Home Care System, which consists of:

- A position tracking system, which localizes Mr. Jones in the apartment.
- An activity tracking system, which monitors the activities carried out by Mr. Jones; activity can take values such as *sitting*, *walking*, *lying*, etc.
- A data monitoring system, in the form of a bracelet, which collects Mr. Jones' basic vital information, such as pulse, skin temperature and skin humidity.
- A person detection system, which is able to detect Mr. Jones, his son and the nurse.
- An emergency monitoring system, identifying emergency situations. This system has a wired connection with the position tracking system, the activity tracking system, the person detection system and the emergency telephony system, and a wireless connection with the data monitoring bracelet.
- An emergency telephony system, which makes emergency calls to Mr. Jones' son in case of emergency.

Assume that neither the nurse nor Mr. Jones' son are located in the apartment, and Mr Jones is walking through a hall of the apartment to his bedroom. He suddenly stumbles, falls down and loses his consciousness, while the data monitoring bracelet that he wears in his wrist is damaged, transmitting erroneous data to the emergency monitoring system.

The emergency telephony system is configured to determine whether it should make an emergency call to his son using the following rule: 'If an emergency situation is detected, and neither the nurse nor Mr. Jones' son are located in the house, then make an emergency call'.

The emergency monitoring system uses the following rules for determining emergency situations: (a) Any abnormal situation is an emergency situation; (b) If Mr. Jones' temperature, skin humidity and pulse have normal values then there is no case of emergency situation; (c) In case Mr. Jones is lying in a place different than his bed, then this is an abnormal situation. Information about Mr. Jones' physical situation is imported from the data monitoring bracelet. In the specific case described above, the bracelet is damaged and keeps erroneously transmitting normal values about Mr. Jones' temperature, skin humidity and pulse. Knowledge about Mr. Jones' current activity (lying) is imported from the activity tracking system, while the position tracking system tracks Mr. Jones' in the hall.

Using information imported from the data monitoring bracelet, and rules a and b, the emergency monitoring system may conclude that this is not an emergency situation. However, based on the information imported from the activity and position tracking systems and rules a and c, the emergency monitoring system reaches the contradictory conclusion that this is an emergency situation. As the data monitoring bracelet is considered more prone to damage than the activity and position tracking systems, and the wired connections between the emergency monitoring system with the activity and position tracking systems are more reliable than the wireless connection with the data monitoring bracelet, the emergency monitoring system determines that this is an emergency situation, and the telephony system reaches the 'emergency call' decision.

2.3. Mushroom Hunting in an Ambient Natural Park

The third scenario takes place in an Ambient Intelligence environment of mushroom hunters, who collect mushrooms in a natural park in North America. The hunters carry mobile devices, which they use to communicate with each other through a wireless network, in order to share their knowledge on edible and non-edible mushrooms.

People interested in picking mushrooms typically do not know every species and family of mushrooms in detail. They know that a deadly mushroom can be very similar to an edible one, e.g., the "*amanita phalloides*" (deadly) and the "*amanita caesarea*" (edible and one of the best mushrooms) that look very much alike. In general, a mushroom hunter has to respect certain rules imposed by the natural park legislation such as the limited quantity of mushrooms that can be picked. Due to the limitation on the allowed quantity, there is the need of establishing the specie of an unknown mushroom during the picking itinerary instead of bringing the picked ones to an expert and discovering, after some days, that the picking has been useless due to the high number of non-edible picked mushrooms. Furthermore, the picking has not been simply useless but it has also vainly cheated the ecosystem of a part. Moreover, even in the case of an irrelevant quantity of non-edible picked mushrooms, it might happen that a small chunk of a deadly mushroom (e.g., "*amanita phalloides*" also known as *The Death Cap*) mixes with edible ones and accidentally eaten. By keeping in mind the above discussed motivations, consider the scenario in which a mushroom hunter, Adam, finds an interesting mushroom but it is unclear if it is edible.

Suppose that the mushroom in question has the following characteristics: It has a stem base featuring a fairly prominent sack that encloses the bottom of the stem (*volva*), and a pale brownish cap with patches, while the margin of the cup is prominently lined, and the mushroom does not have a ring (*annulus*).

Adam has some knowledge on the description of specific species, such as the *Destroying Angel*, the *Death Cap* and the *Caesar's Mushroom*. He also knows that the first two of them are poisonous, while the third one is not. However, the description of the mushroom in question does not fit with any of these species, so Adam cannot determine whether this mushroom is poisonous. He decides to exploit the knowledge of other mushroom hunters in the Ambient Natural Park, and uses the wireless network to contact other hunters that are located nearby. His wireless device establishes connection with the devices of three other hunters.

The first one of the three other hunters, Bob, uses a generic rule, which states that mushrooms with a volva are non-edible. The second hunter, Chris, has knowledge of some specific species that are not toxic, including *springtime amanita*, but does not know their distinct characteristics. The third hunter, Dan, on the other hand, also uses a very generic rule, which states that amanitas are typically dangerous. Using the wireless network, Chris establishes a connection with another hunter, Eric, who knows how *amanita velosa* (a formal name for *springtime amanita*) looks like, and the description of this specific specie fits exactly the description of the mushroom in question.

In this scenario Adam has three options: Using the knowledge of Bob, he will reach the conclusion that the mushroom is poisonous, and therefore he should not pick it. Using the knowledge of Dan, he will reach the same decision. The third option is to use the combined knowledge of Chris and Eric. In the latter case, he will reach a different decision; he will determine that the mushroom is not dangerous, and therefore he may pick it. Being aware that Chris and Eric possess more specialized knowledge than Bob and Dan, he will determine to give priority to the third option determining that the mushroom in question is not poisonous.

2.4. Common Characteristics of the Three Scenarios

The three scenarios described above share some common characteristics with respect to the distribution of context knowledge, the nature of this knowledge, and the relations that exist between the various involved ambient agents. Specifically, the following assumptions have been implicitly made:

– In each case there is an available means of communication through which an ambient agent can communicate and exchange context information with a subset of the other available ambient agents.
– Each ambient agent is aware of the type and quality of knowledge that each of the other agents that it can communicate with possesses, and has specified how part of this knowledge relates to its local knowledge.
– Each ambient agent has some computing and reasoning capabilities that it may use to make certain decisions based on its local and imported context information.
– Each ambient agent is willing to disclose and share part of its local knowledge.

The challenges of reasoning with the available context information and making correct context-dependent decisions in the described scenarios include:

– Local context knowledge may be incomplete, meaning that none of the agents involved in the scenarios described above has immediate access to all the available context information.
– Context knowledge may be ambiguous; in all three scenarios, there is one case that an ambient agent receives mutually inconsistent information from two or more other agents.
– Context knowledge may be inaccurate; e.g. in the first scenario the knowledge about Dr. Amber's schedule possessed by his laptop is not accurate.
– Context knowledge may be erroneous; e.g. in the second scenario, the values for Dr. Jones' temperature, skin humidity and pulse that are transmitted by the bracelet are not valid.
– Each agent may use its own vocabulary to describe its context; e.g. in the third scenario two hunters may use a different name for the same specie of amanita.
– The computational capabilities of most of the devices that are involved in the three scenarios are restricted, so the overhead imposed by the reasoning tasks must not be too heavy.
– The communication load must not also be too heavy, so that the system can quickly reach a decision, taking into account all the available context information that is distributed between the ambient agents. Communication load refers not only to the required number of messages exchanged between the involved devices, but also to the size of these messages.

3. Background & Related Work

3.1. Reasoning Limitations of current Ambient Intelligence systems

The reasoning approaches followed so far in Ambient Intelligence systems either neglect to address the problems caused by the imperfect nature of context, e.g. (Agostini et al., 2005; Sinner et al., 2004; Toninelli et al., 2006; Turhan et al., 2006; Wang, Zhang, Gu and Pung, 2004; Wang, Dong, Chin, Hettiarachchi and Zhang, 2004), or handle them by using heuristics and building additional reasoning mechanisms on top of logic models that cannot inherently deal with the problems of uncertainty, ambiguity and inconsistency. Representative examples of the second category include the context framework of Gaia (Ranganathan and Campbell, 2003) and the Middleware for Context-Aware Mobile Services, SOCAM (Gu et al., 2004). Both of them use first-order predicates for the representation of context information. In Gaia, in order to resolve conflicts that occur when multiple rules are activated in the same time, they have developed a priority base mechanism, allowing only one rule to fire at each time. In SOCAM, to resolve possible conflicts, they have defined sets of rules on the classification and quality information of the context data, considering that different types of context have different levels of confidence, reliability and quality. The development of such priority mechanisms indeed offer some solutions for the problems of uncertainty and ambiguity of context, adding however additional complexity to the reasoning tasks. Moreover, these solutions are rather restricted to meet the needs of the specific systems / applications. For the general needs of Ambient Intelligence systems, a more general and formal approach that can inherently deal with missing, uncertain, inaccurate and ambiguous information is required.

Regarding the distribution of the reasoning tasks, most Ambient Intelligence system have been based on fully centralized architectures. The common approach followed in such systems, e.g. (Chen et al., 2003; Forstadius et al., 2005; Gandon and Sadeh, 2004; Hatala et al., 2005; Kofod-Petersen and Mikalsen, 2005; Patkos et al., 2007; Ranganathan and Campbell, 2003; Toninelli et al., 2006), dictates the existence of a central entity, which is responsible for collecting the available context data from all sensors and ambient agents operating in the same environment, and for all the required reasoning tasks, which may include transforming the imported context data in a common format, deducing higher-level context information form the raw context data, and taking context-dependent decisions for the behavior of the system. The need for more decentralized approaches has recently led several research teams to deploy methods and techniques from Distributed Artificial Intelligence, with the *shared memory* or *blackboard* based architectures of (Khushraj et al., 2004; Krummenacher et al., 2005; Korpipaa et al., 2003) being the most prominent examples. Collecting the reasoning tasks in a central entity certainly has many advantages; it can achieve better control and better coordination between the various entities that have access to the central entity. Blackboard-based and shared-memory models have been thoroughly studied and used in many different types of distributed systems and have proved to work well in practice. The requirements are, though, much different in this setting. Context may not be restricted to a small room, office or apartment; cases of broader areas must also be considered. The communication with a central entity is not always guaranteed, and wireless communications are typically unreliable and restricted by the range of the transmitters. Thus, a fully distributed scheme seems to be a necessity.

3.2. Multi-Context Systems

A Multi-Context System consists of a set of *contexts* and a set of inference rules (known as *mapping* or *bridge* rules) that enable information flow between different contexts. A context can be thought of as a logical theory - a set of axioms and inference rules - that models local knowledge of an agent. Different contexts are expected to use different languages and inference systems, and although each context may be locally consistent, global consistency cannot be required or guaranteed. Reasoning with multiple contexts requires performing two types of reasoning; (a) *local reasoning*, based on the individual context theories; and (b) *distributed reasoning*, which combines the knowledge encoded in different local theories using the mappings. The most critical challenges of contextual reasoning are the *heterogeneity* of local context theories, and the potential conflicts that may arise from the interaction of different contexts through the mappings.

The notions of *context* and *contextual reasoning* were first introduced in AI by McCarthy in (1987), as an approach for the problem of *generality.* In the same paper, he argued that the combination of non-monotonic reasoning and contextual reasoning would constitute an adequate solution to this problem. Since then, two main formalizations have been proposed to formalize context: the Propositional Logic of Context, *PLC* (Buvac and Mason, 1993; McCarthy and Buvač, 1998), and the Multi-Context Systems introduced in (Giunchiglia and Serafini, 1994), which later became associated with the Local Model Semantics proposed in (Ghidini and Giunchiglia, 2001). Multi-Context Systems have been argued to be most adequate with respect to the three properties of contextual reasoning (*partiality*, *approximation*, *proximity*) and shown to be technically more general than PLC (Serafini and Bouquet, 2004). This formalism was also the basis of two recent studies that were the first to deploy non-monotonic features in contextual reasoning:

1. the non-monotonic rule-based MCS framework (Roelofsen and Serafini, 2005), which supports default negation in the mapping rules allowing to reason based on the absence of context information
2. the multi-context variant of Default Logic, *ConDL* (Brewka et al., 2007), which models bridge relations between different contexts as *default rules.*

Additionally to the three fundamental dimensions of contextual reasoning (partiality, approximation and perspective) that the generic MCS model inherently supports, both approaches support reasoning with incomplete local information using default negation in the body of the mapping rules. Furthermore, Contextual Default Logic handles the problem of mutually inconsistent information provided by two or more different sources using default mapping rules, and has the additional advantage that is closer to implementation due to the well-studied relation between Default Logic and Logic Programming. However, *ConDL* does not provide ways to model the quality of imported knowledge, nor preference between different information sources, leaving the conflicts that arise in such cases unresolved.

The use of Multi-Context Systems as a means of specifying and implementing agent architectures has been proposed in some recent studies. Specifically, (Sabater et al., 2002; Casali et al., 2008) 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. A similar approach is followed in (Dastani et al., 2007), where *contextual deliberation* of cognitive agents is achieved using a special extension of Defeasible Logic. On the other hand, in the multi-agent architectures proposed in (Cristani and Burato, 2009; Resconi and Kovalerchuk, 2009), context refers to a criterion, with respect to which an agent thinks it is important to evaluate an action. In our case, a context represents the viewpoint of each different agent in the system.

3.3. Peer Data Management Systems

Peer data managements systems can be viewed as special cases of Multi-Context Systems, as they consist of autonomous logic-based entities (*peers*) that exchange local information using *bridge rules*. A key issue in formalizing data-oriented Peer-to-Peer systems is the semantic characterization of *mappings* (bridge rules). One approach, followed in (Bernstein et al., 2002; Halevy et al., 2003), is the first-order logic interpretation of Peer-to-Peer systems. (Calvanese et al., 2004) identified several drawbacks with this approach, regarding modularity, generality and decidability, and proposed new semantics based on epistemic logic. A common problem of both approaches is that they do not model and thus cannot handle inconsistency. (Franconi et al., 2003) extended the autoepistemic semantics to formalize local inconsistency. The latter approach guarantees that a locally inconsistent database base will not render the entire knowledge base inconsistent. A broader extension, proposed in (Calvanese et al., 2005), is based on non-monotonic epistemic logic, and enables isolating local inconsistency, while also handling peers that may provide mutually inconsistent data. It guarantees that in case of importing knowledge that would render the local knowledge inconsistent, the local peer knowledge base remains consistent by discarding a minimal amount of the data retrieved from the other peers. The propositional Peer-to-Peer Inference System proposed in (Chatalic et al., 2006) extends the distributed reasoning methods of (Adjiman et al., 2006) to deal with conflicts caused by mutually inconsistent information sources, by detecting them and reasoning without them. Finally, based on the latter study, (Binas and McIlraith, 2007) proposes algorithms for inconsistency resolution in Peer-to-Peer Query Answering exploiting a preference relation on the peers.

The three latter approaches ((Calvanese et al., 2005), (Chatalic et al., 2006) and (Binas and McIlraith, 2007)), have some common characteristics that meet many of the requirements of Ambient Intelligence environments that we discuss in Section 2. Specifically, they support information flow between different agents through mapping rules, enable reasoning with incomplete local information, and handle (each one in its own way) agents that provide mutually inconsistent information. However, regarding their deployment in Ambient Intelligence, they have the following limitations:

- The approach of (Calvanese et al., 2005) assumes that all peers share a common alphabet of constants, which is not always realistic in ambient environments.
- The approaches of (Calvanese et al., 2005) and (Chatalic et al., 2006) do not

include the notion of preference between system peers, which could be used to resolve potential conflicts caused by mutually inconsistent information sources.
- The method followed by (Binas and McIlraith, 2007) assumes a global preference relation on the systems peers, which is shared and used by all peers. This feature is in contrast with the dimension of perspective, which allows each agent to use its own preference relation based on its own viewpoint.
- The distributed algorithms used in (Chatalic et al., 2006) and (Binas and McIlraith, 2007) assume that the inconsistencies caused by the mappings of a newly joined peer must be computed at the time the mappings are created, and not at reasoning time. This has two implications: (a) It may produce an additional possibly unnecessary computational overhead to a peer, considering that it may never have to use this information; (b) This information may become stale, in the sense that some of the mappings that cause the inconsistencies may have been defined by a peer which has left the system at the time of query evaluation.
- The studies of (Chatalic et al., 2006) and (Binas and McIlraith, 2007) do not deal with cases of local inconsistency.
- None of the approaches include the notion of privacy. All peers are expected to cooperate and share their local knowledge with all other system peers.

4. Context Representation Model

We model a MCS 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 ordering 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 a local rule is a conjunction of *local* literals (literals that are contained in V_i), while its head contains a local literal. There are two types of local rules:

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

Mapping rules associate literals from the local vocabulary V_i (*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. Mapping rules are modeled as defeasible rules of the form:

$$r_i^m : a_i^1, a_j^2, ...a_k^{n-1} \Rightarrow a_i^n$$

r_i^m associates local literals of C_i (e.g. a_i^1) with local literals of C_j (a_j^2), C_k (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 strict total preference ordering T_i on C to express its confidence on the knowledge it imports from other contexts. This is of the form:

$$T_i = [C_k, C_l, ..., C_n]$$

According to T_i, C_k is preferred to C_l by C_i, if the rank of C_k is lower than the rank of C_l in T_i. The strict total preference ordering enables resolving all potential conflicts that may arise from the interaction of contexts through their mapping rules. An alternative choice, which is closer to the needs of ambient applications, is partial ordering. However, this would add complexity to the reasoning tasks, and would enable resolving certain conflicts only.

We have deliberately chosen to use the simplest version of Defeasible Logic, and disregard *facts*, *defeaters* and the *superiority relation* between rules, which are used in fuller versions of Defeasible Logic (Antoniou et al., 2001), to keep the discussion and technicalities simple. Besides, the results of (Antoniou et al., 2001) have shown that these elements can be simulated by the other ingredients of the logic. We should also note that the proposed model may also support overlapping vocabularies and enable different contexts to use elements of common vocabularies (e.g. URIs) by adding a context identifier, e.g. as a prefix in each such word, adding the modified words in the vocabularies of the contexts, and using appropriate mappings to associate them.

In the model that we described, two types of conflicts may arise: (a) *local conflicts*, caused by local rules with complementary conclusions; and (b) *global conflicts*, caused by the interaction of contexts through their mapping rules. Local conflicts are resolved according to the Proof Theory of Defeasible Logic (Antoniou et al., 2001). Specifically, whenever two different chains of rules lead to contradictory conclusions, to determine which is the valid conclusion, the chains are compared with respect to the types of rules they contain. Chains of strict rules dominate over chains that also contain defeasible rules, while between two chains that contain at least one defeasible rule, neither of them dominates. On the other hand, global conflicts resolution takes into account the preference ordering on the system contexts. In this case, the *strength* of a rule r_i with respect to a context C_i depends on the preference ranks in T_i of the contexts that the literals in the body of r_i are defined by. In the next two sections, the overall reasoning model and four alternative strategies for conflict resolution are described in detail through four versions of a distributed algorithm for query evaluation.

Example 1. The representation model described above is applied as follows to the scenario described in Section 2. The local knowledge of the mobile phone (denoted as C_1) is encoded in strict local rules $r_{1,1}^l$ and $r_{1,2}^l$, while local defeasible rules $r_{1,3}^d$ and $r_{1,4}^d$ encode Dr. Amber's preferences.

$r_{1,1}^l :\rightarrow incoming\_call_1$

$r^l_{1,2} : \rightarrow normal\_mode_1$
$r^d_{1,3} : incoming\_call_1, \neg lecture_1 \Rightarrow ring_1$
$r^d_{1,4} : silent\_mode_1 \Rightarrow \neg ring_1$

Mapping rules $r^m_{1,5}$ and $r^m_{1,6}$ encode the associations of the local knowledge of the mobile phone with the context knowledge of Dr. Amber's laptop (C_2), the localization service (C_3), and the classroom manager (C_4).

$r^m_{1,5} : classtime_2, location\_RA201_3 \Rightarrow lecture_1$
$r^m_{1,6} : \neg class\_activity_4 \Rightarrow \neg lecture_1$

The local context knowledge of the laptop (C_2), the localization service (C_3), the classroom manager (C_4) and the person detection service (C_5) is encoded in rules $r^l_{2,1}$, $r^l_{3,1}$, $r^l_{4,1}$ and $r^l_{5,1}$, respectively. To import information from the person detection service, the classroom manager uses rule $r^m_{4,2}$.

$r^l_{2,1} : \rightarrow classtime_2$
$r^l_{3,1} : \rightarrow location\_RA201_3$
$r^l_{4,1} : \rightarrow projector(off)_4$
$r^m_{4,2} : projector(off)_4, detected(1)_5 \Rightarrow \neg class\_activity_4$
$r^l_{5,1} : \rightarrow detected(1)_5$

The mobile phone is configured to give highest priority to information imported by the classroom manager and lowest priority to the person detection service. This is described in preference order $T_1 = [C_4, C_3, C_2, C_5]$.

5. Distributed Query Evaluation

This section provides an operational model in the form of a distributed algorithm for query evaluation in MCS following the model described in Section 4. The first subsection describes the algorithm in detail and explains how it works through an example, while the second one studies its formal properties.

5.1. Algorithm Description

$P2P\_DR$ is a distributed algorithm for query evaluation in Multi-Context Systems following the model described in Section 4. The specific reasoning 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:

– *true*; indicating that p_i is a logical conclusion in C
– *false*; indicating that p_i is not a logical conclusion in C
– *undefined*; indicating that we cannot prove whether p_i is a logical conclusion in C

$P2P\_DR$ proceeds in four main steps. In the first step (lines 1-8 in the pseudocode given below), $P2P\_DR$ determines whether p_i, or its negation $\sim p_i$ are consequences of the local strict rules of C_i, using *local_alg*, a local reasoning algorithm, which is described later in this section. If *local_alg* computes *true* as an answer for p_i or $\sim p_i$, $P2P\_DR$ returns *true/false* respectively as an answer for p_i and terminates.

In the second step (lines 9-12), $P2P\_DR$ calls *Support* (described later in this section) 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* also returns two data structures for p_i: (a) the set of foreign literals used in the most preferred (according to T_i) chain of applicable rules for p_i (SS_{p_i}); and (b) 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, the algorithm returns *false* as an answer and terminates.

In the third step (lines 13-14), $P2P\_DR$ calls *Support* to compute the respective constructs for $\sim p_i$ ($SS_{\sim p_i}$, $BS_{\sim p_i}$).

In the last step (lines 15-24), $P2P\_DR$ uses the constructs computed in the previous steps and the preference order defined by C_i (T_i), to determine the truth value of p_i. In case there is no unblocked rule for $\sim p_i$ ($unb_{p_i} = false$), or SS_{p_i} is computed by *Stronger* (described later in this section) 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 $BS_{\sim p_i}$ 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 ($[p_1, ..., p_i]$)
- 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**($p_i, C_0, C_i, Hist_{p_i}, T_i, SS_{p_i}, BS_{p_i}, Ans_{p_i}$)
1: call $local\_alg(p_i, localAns_{p_i})$
2: **if** $localAns_{p_i} = true$ **then**
3:     $Ans_{p_i} \leftarrow true$, $SS_{p_i} \leftarrow \emptyset$, $BS_{p_i} \leftarrow \emptyset$
```

```
4:     terminate
5: call $local\_alg(\sim p_i, localAns_{\sim p_i})$
6: **if** $localAns_{\sim p_i} = true$ **then**
7:     $Ans_{p_i} \leftarrow false$, $SS_{p_i} \leftarrow \emptyset$, $BS_{p_i} \leftarrow \emptyset$
8:     terminate
9: call $Support(p_i, Hist_{p_i}, T_i, sup_{p_i}, unb_{p_i}, SS_{p_i}, BS_{p_i})$
10: **if** $unb_{p_i} = false$ **then**
11:     $Ans_{p_i} \leftarrow false$, $SS_{p_i} \leftarrow \emptyset$, $BS_{p_i} \leftarrow \emptyset$
12:     terminate
13: $Hist_{\sim p_i} \leftarrow (Hist_{p_i} - \{p_i\}) \cup \{\sim p_i\}$
14: call $Support(\sim p_i, Hist_{\sim p_i}, T_i, sup_{\sim p_i}, unb_{\sim p_i}, SS_{\sim p_i}, BS_{\sim p_i})$
15: **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**
16:     $Ans_{p_i} \leftarrow true$
17:     **if** $C_0 \neq C_i$ **then**
18:         $SS_{p_i} \leftarrow \emptyset$, $BS_{p_i} \leftarrow \emptyset$
19: **else if** $sup_{\sim p_i} = true$ and $Stronger(BS_{p_i}, SS_{\sim p_i}, T_i) \neq BS_{p_i}$ **then**
20:     $Ans_{p_i} \leftarrow false$, $SS_{p_i} \leftarrow \emptyset$, $BS_{p_i} \leftarrow \emptyset$
21: **else**
22:     $Ans_{p_i} \leftarrow undefined$
23:     **if** $C_0 \neq C_i$ **then**
24:         $SS_{p_i} \leftarrow \emptyset$, $BS_{p_i} \leftarrow \emptyset$
```

local_alg is called by *P2P_DR* to determine whether the truth value of the queried literal can be derived from the local strict rules of a context theory (R^s). We should note again 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($p_i, localAns_{p_i}$)

```
1: **for all** $r_i \in R^s[p_i]$ **do**
2:     **for all** $b_i \in body(r_i)$ **do**
3:         call $local\_alg(b_i, localAns_{b_i})$
4:     **if** for all $b_i$: $localAns_{b_i} = true$ **then**
5:         **return** $localAns_{p_i} = true$ and terminate
6: **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 $sup_{p_i} = true$; otherwise, it returns $sup_{p_i} = false$. 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 ($[p_1, ..., p_i]$)
– 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($p_i, Hist_{p_i}, T_i, sup_{p_i}, unb_{p_i}, SS_{p_i}, BS_{p_i}$)
1: $sup_{p_i} \leftarrow false$
2: $unb_{p_i} \leftarrow false$
3: for all $r_i \in R[p_i]$ do
4:    $cycle(r_i) \leftarrow false$
5:    $SS_{r_i} \leftarrow \emptyset$
6:    $BS_{r_i} \leftarrow \emptyset$
7:    for all $b_t \in body(r_i)$ do
8:       if $b_t \in Hist_{p_i}$ then
9:          $cycle(r_i) \leftarrow true$
10:         $BS_{r_i} \leftarrow BS_{r_i} \cup \{d_t\}$ {$d_t \equiv b_t$ if $b_t \notin V_i$; otherwise $d_t$ is the first foreign
            literal of $C_i$ added in $Hist_{p_i}$ after $b_t$}
11:      else
12:         $Hist_{b_t} \leftarrow Hist_{p_i} \cup \{b_t\}$
13:         call $P2P\_DR(b_t, C_i, C_t, Hist_{b_t}, T_t, SS_{b_t}, BS_{b_t}, Ans_{b_t})$
14:         if $Ans_{b_t} = false$ then
15:            stop and check the next rule
16:         else if $Ans_{b_t} = undefined$ or $cycle(r_i) = true$ then
17:            $cycle(r_i) \leftarrow true$
18:            if $b_t \notin V_i$ then
19:               $BS_{r_i} \leftarrow BS_{r_i} \cup \{b_t\}$
20:            else
21:               $BS_{r_i} \leftarrow BS_{r_i} \cup BS_{b_t}$
22:         else
23:            if $b_t \notin V_i$ then
24:               $BS_{r_i} \leftarrow BS_{r_i} \cup \{b_t\}$
25:               $SS_{r_i} \leftarrow SS_{r_i} \cup \{b_t\}$
```

```
26:         else
27:           $BS_{r_i} \leftarrow BS_{r_i} \cup BS_{b_t}$
28:           $SS_{r_i} \leftarrow SS_{r_i} \cup SS_{b_t}$
29:   if $unb_{p_i} = false$ or $Stronger(BS_{r_i}, BS_{p_i}, T_i) = BS_{r_i}$ then
30:     $BS_{p_i} \leftarrow BS_{r_i}$
31:   $unb_{p_i} \leftarrow true$
32:   if $cycle(r_i) = false$ then
33:     if $sup_{p_i} = false$ or $Stronger(SS_{r_i}, SS_{p_i}, T_i) = SS_{r_i}$ then
34:       $SS_{p_i} \leftarrow SS_{r_i}$
35:     $sup_{p_i} \leftarrow true$
```

The $Stronger(A, B, T_i)$ 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 C_k (the context that defines a_k) has lower rank than C_l (the context that defines b_l) in T_i. The strength of a set is determined by the the *weakest* (least preferred) literal in this set.

Stronger(A, B, T_i)

```
1: if $\exists b_l \in B$: $\forall a_k \in A$: $C_k$ has lower rank than $C_l$ in $T_i$
   or ($A = \emptyset$ and $B \neq \emptyset$) then
2:   $Stronger = A$
3: else if $\exists a_k \in A$: $\forall b_l \in B$: $C_l$ has lower rank than $C_k$ in $T_i$
   or ($B = \emptyset$ and $A \neq \emptyset$) then
4:   $Stronger = B$
5: else
6:   $Stronger = None$
```

Example 1 (continued) In the MCS of example 1, given a query about $ring_1$, $P2P\_DR$ proceeds as follows:

– In the first step, $P2P\_DR$ fails to compute an answer for $ring_1$ based on the local strict rules of C_1.

– In the second step, it calls *Support* for $ring_1$. r^d_{13} is the only rule with head $ring_1$. *Support* calls $P2P\_DR$ for $incoming\_call_1$ and $\neg lecture_1$, which are the literals in the body of r^d_{13}.

– Using rule r^l_{11}, $local\_alg$ computes *true* as a local answer for $incoming\_call_1$ and therefore $P2P\_DR$ returns $Ans_{incoming\_call_1} = true$.

– $P2P\_DR$ fails to compute an answer for $\neg lecture_1$ based on the local strict rules of C_1, so it calls *Support* for $\neg lecture_1$. r^m_{16} is the only rule with head $\neg lecture_1$. *Support* calls $P2P\_DR$ for $\neg class\_activity_4$, which is the only literal in the body of r^m_{15}.

– $P2P\_DR$ fails to compute an answer for $\neg class\_activity_4$ based on the local strict rules of C_4, so it calls *Support* for $\neg class\_activity_4$. r^m_{42} is the only rule with head $\neg class\_activity_4$. *Support* calls $P2P\_DR$ for $projector(off)_4$ and $detected(1)_5$, which are the literals in the body of r^m_{42}.

– Using rule r^l_{41}, $local\_alg$ computes *true* as a local answer for $projector(off)_4$ and therefore $P2P\_DR$ returns $Ans_{projector(off)_4} = true$.

– Using rule r^l_{51}, $local\_alg$ computes *true* as a local answer for $detected(1)_5$ and therefore $P2P\_DR$ returns $Ans_{detected(1)_5} = true$.

- *Support* computes $SS_{r^m_{42}} = \{detected(1)_5\}$, and since there is no rule with head $class\_activity_4$, $P2P\_DR$ returns $Ans_{\neg class\_activity_4} = true$.
- *Support* computes $SS_{r^m_{16}} = \{\neg class\_activity_4\}$, and since there is no other rule with head $\neg lecture_1$, $BS_{\neg lecture_1} = SS_{\neg lecture_1} = \{\neg class\_activity_4\}$.
- In the next step, $P2P\_DR$ calls *Support* for $lecture_1$. $lecture_1$ is in the head of rule r^m_{15}, hence *Support* calls $P2P\_DR$ to compute the truth values of $classtime_2$ and $location\_RA201_3$, which are contained in the body of r^m_{15}.
- Both $classtime_2$ and $location\_RA201_3$ are derived as conclusions of the strict local rules in C_2 and C_3 respectively; hence for both literals $P2P\_DR$ returns *true* as an answer.
- *Support* computes $SS_{r^m_{15}} = \{classtime_2, location\_RA201_3\}$, and since there is no other rule with head $lecture_1$, $BS_{lecture_1} = \{classtime_2, location\_RA201_3\}$.
- Using $T_1 = [C_4, C_3, C_2, C_5]$, $P2P\_DR$ determines that $SS_{\neg lecture_1}$ is stronger than $BS_{lecture_1}$, and returns $Ans_{\neg lecture_1} = true$.
- *Support* computes $SS_{r^d_{13}} = \{\neg class\_activity_4\}$, and since there is no other rule with head $lecture_1$, it returns $BS_{ring_1} = SS_{ring_1} = \{\neg class\_activity_4\}$.
- In the next step, $P2P\_DR$ calls *Support* for $\neg ring_1$. $\neg ring_1$ is in the head of rule r^d_{14}, hence *Support* calls $P2P\_DR$ for literal $silent\_mode_1$, which is the only literal in the body of r^d_{13}. Since there is no rule with head $silent\_mode_1$, $P2P\_DR$ returns $Ans_{silent\_mode_1} = false$.
- As there is no other rule with $\neg ring_1$ in its head, *Support* returns $unb_{\neg ring_1} = false$, and eventually $P2P\_DR$ returns $Ans_{ring_1} = true$.

5.2. Properties of the Algorithm

Below, we describe some formal properties of $P2P\_DR$ regarding its termination, complexity, and the possibility to create an equivalent unified defeasible theory from the distributed context theories.

5.2.1. Termination

Proposition 1 refers to the termination of $P2P\_DR$, and is a consequence of the cycle detection process within the algorithm.

Proposition 1. $P2P\_DR$ terminates in finite time returning one of the values *true*, *false* and *undefined* as an answer for the queried literal.

Proof. At each recursive call, $P2P\_DR$ makes at most two calls of $local\_alg$ (for p_i and $\sim p_i$), two calls of *Support* (for p_i and $\sim p_i$) and two calls of $Stronger$.

local_alg checks the local answers for all literals in the bodies of all strict local rules with head p_i (or $\sim p_i$). By definition, all such rules are defined by context C_i, and are finite in number. Since V_i (the vocabulary of C_i) is a finite set of literals, each local rule contains a finite set of literals in its body. Therefore, one call of *local_alg* induces a finite number of operations. Since, one type of such operations involves a recursive call of *local_alg*, we also have to prove that the total number of recursive calls of *local_alg* is not infinite. Assume that one call of *local_alg* induces infinite recursive calls of *local_alg*. Since we have assumed that there are no loops in a context theory, each such call would be for a different literal in V_i. However, by the fact that there is a finite number of literals in

V_i, the total number of recursive calls of *local_alg* is bounded by the number of literals in V_i. Therefore, no call of *local_alg* can induce infinite recursive calls of *local_alg*. Consequently, *local_alg* terminates in finite time returning either *true* or *false* as a local answer for the queried literal.

Support checks the answers for all literals in the bodies of all rules with head p_i (or $\sim p_i$). By definition, all such rules are defined by context C_i, and are finite in number. Since each literal in the bodies of these rules is in the vocabulary V_j of a context $C_j \in C$, and by the facts that there is a finite number of contexts in C, and that each vocabulary is a finite set of literals, each such rule contains a finite set of literals in its body. Therefore, one call of *Support* induces a finite number of checking operations. Since, one type of such operations involves a recursive call of *P2P_DR*, we also have to prove that the total number of recursive calls of *P2P_DR* is not infinite. Assume that one call of *P2P_DR* induces through *Support* infinite recursive calls of *P2P_DR* and *Support*. At each recursive call, the structure that keeps track of the *history* of the query ($Hist$) is augmented with a literal q_j, where q_j belongs to the vocabulary V_j of a context $C_j \in C$, and q_j is not already contained in $Hist$. As the total number of contexts in C is finite, and each vocabulary is a finite set of literals, the total number of recursive calls of *P2P_DR* and *Support* is bounded by the total number of literals in $V = \bigcup V_i$. Therefore, no call of *Support* can induce infinite recursive calls of *P2P_DR* and *Support*. Since, *Support* additionally induces at most two calls of *Stronger*, we also have to prove that *Stronger* also terminates in finite time.

Stronger requires checking the preference ranks of the contexts that have defined the literals contained in two Supportive/Blocking Sets. A Supportive / Blocking Set is a set of literals derived by contexts in C. Since, we have already proved that there is a finite number of literals defined in C, each such set contains a finite number of elements. Therefore, given two sets A and B, and a preference order T_i, *Stronger* terminates in finite time, returning either A, B or *none*.

Consequently, since *local_alg*, *Support* and *Stronger* terminate in finite time, *P2P_DR* also terminates in finite time. By definition of the algorithm, it is trivial to verify, that one of the values *true*, *false* and *undefined* is returned as an answer for p_i upon termination.

5.2.2. Complexity

In this section we provide a complexity analysis in terms of *computational complexity* of the proposed algorithms, and *number of algorithm calls* and *number of messages* imposed by distributed query evaluation.

Computational Complexity

The term *computational complexity* refers to the total number of operations imposed by a single call of the algorithms. Obviously, the complexity of the local algorithm, *local_alg*, is related only to the strict local rules and local literals of a context. By definition of *local_alg* and by the fact that there are no loops in the local context theories, it is trivial to prove that the complexity of *local_alg* is proportional to the number of strict local rules of a context, and to the total number of its local literals.

The complexity of $Stronger(A, B, T_i)$ is related to the total number of elements contained in the two sets, A and B. *Stronger* can be implemented in a slightly different way than that described in the pseudocode provided in the previous subsection. Specifically, it requires a process that identifies the *weakest* literal in each of the two sets based on the ranks of the contexts that the literals are defined by, and a comparison of the *strength* of the weakest literals. The complexity of this process is proportional to the total number of elements contained in each set.

Support imposes one operation for each of the literals contained in each rule that contains the queried literal or its negation in its head. It also requires computing the *strongest* between the Supportive or Blocking Sets of all such rules. Taking into account the complexity of *Stronger*, and by the fact that, in the worst case, all rules of a context may be relevant, the complexity of *Support* is in the worst case proportional to the number of rules of a context theory and to the number of literals defined in the system.

By definition, $P2P\_DR$ calls *local_alg*, *Support* and *Stronger* twice. Taking into account their complexities, we obtain the following result for the complexity of $P2P\_DR$:

Proposition 2. The number of operations imposed by one call of $P2P\_DR$ for the evaluation of a query for literal p_i is, in the worst case that all rules of C_i contain either p_i or $\sim p_i$ in their head and all literals defined in the system in their bodies, proportional to the number of rules in C_i, r_i, and to the total number n of literals defined in the system ($O(r_i, n)$).

Algorithm Calls & and Number of Messages

In order to reduce the total number of algorithm calls and messages that $P2P\_DR$ may impose, we modify the original algorithms as follows. For each context C_i, we use two structures that the algorithms may access to store or retrieve the results obtained during a query evaluation process:

- OUT_Q, which stores the results of queries for foreign literals of C_i that are contained in the bodies of mapping rules of C_i
- INC_Q, which stores the results of queries for local literals of C_i

OUT_Q contains records of the form $rec(p_j, Hist_{p_j}) : Ans_{p_j}$, where $p_j \notin V_i$, and $Hist_{p_i}$ is a set of local or foreign literals of C_i. Each such record contains Ans_{p_j} for a query for literal p_j with history $Hist_{p_j}$, which has already been evaluated during the same query evaluation process.

INC_Q contains records of the form $rec(p_i, Hist_{p_i}) : (Ans_{p_i}, BS_{p_i}, SS_{p_i})$, where $p_i \in V_i$ and $Hist_{p_i}$ is a set of local or foreign literals of C_i. Each record represents the results that have already been obtained during the same query evaluation process for a query for literal p_i with history $Hist_{p_i}$. The results contained in one record include Ans_{p_i}, and (in case $Ans_{p_i} \neq false$) BS_{p_i} and SS_{p_i}. For each local literal p_i, INC_Q contains an additional record of the form $rec(p_i) : localAns_{p_i}$ that retains $localAns_{p_i}$, which is independent of the history of the query as we have assumed that there are no loops in the local context theories.

Using these two structures, $P2P\_DR$ and *Support* are modified as follows. Before calling *local_alg* to compute the local answers for p_i and $\sim p_i$, $P2P\_DR$ checks whether there are already records for p_i in INC_Q, In case there are such records, it retrieves the local answers for p_i and $\sim p_i$ from INC_Q. Otherwise, after evaluating the local answers using *local_alg*, $P2P\_DR$ creates appropriate records for p_i and $\sim p_i$ in INC_Q. Also, after evaluating Ans_{p_i}, BS_{p_i} and SS_{p_i}

using *Support* and *Stronger*, $P2P\_DR$ creates an appropriate record for the query for p_i with history $Hist_{p_i}$ in INC_Q.

Support is also modified as follows. Before calling $P2P\_DR$ to compute Ans_{b_t}, SS_{b_t} and BS_{b_t} for each of the literals b_t contained in the body of the rules with head p_i, *Support* checks whether there is a record for b_t and $Hist_{b_t}$ in INC_Q or OUT_Q. In case there is no such record, it calls $P2P\_DR$ and creates an appropriate record for (b_t,$Hist_{b_t}$) in INC_Q if b_t is a local literal of C_i, or in OUT_Q otherwise.

We should note that since we assume that the state of the network remains unchanged during a query evaluation process, but may change in the meantime between two consecutive query evaluation processes, the two structures INC_Q and OUT_Q are updated every time a new query is posed to the system. The following proposition refers to the complexity of query evaluation using the optimized versions of $P2P\_DR$ and *Support*.

Proposition 3. The total number of calls of $P2P\_DR$ that are required for the evaluation of a single query is in the worst case $O(n \times \sum P(n,k))$, where n stands for the total number of literals in the system, $\sum$ expresses the sum over $k = 0, 1, ..., n$, and $P(n,k)$ stands for the number of permutations with length k of n elements. If each of the literals in the system is defined by a different context, then the total number of messages exchanged between the system contexts for the evaluation of a query is $O(2 \times n \times \sum P(n,k))$.

Proof. In the worst case in a distributed query evaluation process with $P2P\_DR$, each context has to call $P2P\_DR$ once for each of the literals that appear in its local theory and mappings and for each different *history* of the query. Assume that all contexts use all system literals in their theories. Obviously, two different calls of $P2P\_DR$ for a literal p_j with same query *history* cannot be made by two different contexts. This means, that overall for each literal in the system and for each different *history* of a query for that literal, at most one (and in the worst case exactly one) call of $P2P\_DR$ will be made. Hence, the total number of calls of $P2P\_DR$ will be proportional to the total number of system literals, and to the number of different *histories* that may appear in one call. One query *history* is actually a permutation of a subset of the system literals. One literal cannot appear more than once in a query *history*, as in this case $P2P\_DR$ will have already detected a loop (*cycle*), and will not permit a recursive call. Since, the number of different permutations of the system literals is equal to $\sum P(n,k))$, where n stands for the total number of system literals, $\sum$ expresses the sum over $k = 0, 1, ..., n$, and $P(n,k)$ stands for the number of permutations with length k of n elements, the total number of calls of $P2P\_DR$ is $O(n \times \sum P(n,k))$. Regarding the number of messages, if each literal is defined by a different context, then each call of $P2P\_DR$ actually is implemented as a query message between two different contexts. Taking into account also the response messages, the total number of messages for the evaluation of a query is $O(2 \times n \times \sum P(n,k))$.

We should note that $\sum P(n,k)$ stands between 2^n and $n!2^n$. If we assume that there are no loops in the global knowledge base (acyclic MCS), then the history of a query is irrelevant to how it is evaluated. In this case, INC_Q and OUT_Q retain at most one record for each of the local and foreign literals of a context, and the complexity of query evaluation is reduced as follows.

Proposition 4. In acyclic MCS, the total number of calls of $P2P\_DR$ that are required for the evaluation of a single query is in the worst case $O(c \times n)$, where c stands for the total number of contexts in the system, and n stands for the total number of literals in the system. If each of the literals in the system is defined by a different context, then the total number of messages exchanged between the system contexts for the evaluation of a query is $O(2 \times c \times n)$.

Proof. In acyclic MCS, there are no loops in the global knowledge base, and therefore there is no case for $P2P\_DR$ to detect a cycle during query evaluation. In this case, the process and outcome of a query evaluation do not depend on the *history* of the query. Therefore, for $P2P\_DR$ we need structures for incoming and outgoing queries with only one record for each of the literals that a query about them has already been evaluated. In the worst case, that each context uses all system literals in its mappings, and the evaluation of a query involves all mapping rules from all contexts, $P2P\_DR$ is called at most (and in the worst case exactly) once by each context for each literal in the system. Therefore the number of calls of $P2P\_DR$ is in the worst case $c \times n$, where c is the total number of contexts and n is the total number of literals in the system. Regarding the number of messages, if each literal is defined by a different context, then each call of $P2P\_DR$ actually is implemented as a query message between two different contexts. Taking into account also the response messages, the total number of messages for the evaluation of a query is in the worst case $2 \times c \times n$.

5.2.3. Equivalent Global Defeasible Theory

The goal of the procedure that we describe in this section is the construction of a global defeasible theory $T_v(C)$, which produces the same results as the application of $P2P\_DR$ on a Multi-Context System C under the proof theory of the ambiguity blocking version of Defeasible Logic with superiority relation (Antoniou et al., 2001). The existence of this procedure enables resorting to centralized reasoning by collecting the distributed context theories in a central entity and creating an equivalent defeasible theory. In addition, this result is typical of other works in the area of Peer-to-Peer reasoning, in which the distributed query evaluation algorithm is related to querying a single knowledge base that can be constructed, e.g. (Adjiman et al., 2006). Via Proposition 5, $P2P\_DR$ has a precise semantic characterization. Defeasible Logic has a proof-theoretic (Antoniou et al., 2001), an argumentation-based (Governatori et al., 2004) and a model-theoretic semantics (Maher, 2002). The result holds for acyclic MCS and is described in Proposition 5.

Proposition 5. For a literal p_i in V_i, $P2P\_DR$ computes
(1) $Ans_{p_i} = true$ iff $T_v(C) \vdash +\partial p_i$
(2) $Ans_{p_i} = false$ iff $T_v(C) \vdash -\partial p_i$

In the rest of the section, we provide a proof sketch for Proposition 5 by describing in detail the steps of the procedure that constructs $T_v(C)$.

Proof Sketch. The procedure follows three main steps:

1. The local strict rules of each context theory are added as strict rules in $T_v(C)$.
2. The local defeasible and mapping rules of each context theory are added as defeasible rules in $T_v(C)$.
3. For each pair of rules with contradictory conclusions, a priority relation is added taking into account the preference orderings of the system contexts.

The vocabulary used by $T_v(C)$ (V) is the union of the vocabularies of the unified context theories: $V = \bigcup V_i$. The construction of the superiority relation of the global defeasible theory is achieved using the *Priorities* process described below. The role of this process is to augment $T_v(C)$, as this is derived from the first two steps of the procedure described above, with the additional required rule priorities considering the preference orderings of the system contexts.

Priorities

The derivation of priorities between competing rules (rules with contradictory conclusions) in $T_v(C)$ is a finite sequence $Pr = (Pr(1), ..., Pr(n))$, where each $Pr(i)$ can be one of the followings:

– The Supportive Set of a rule in $T_v(C)$ (a set of literals).
– A priority relation between two conflicting rules in $T_v(C)$
– The Supportive Set of a literal in $T_v(C)$ (a set of literals).

In this process we use two special elements: (a) w, to mark the rules that cannot be applied to support their conclusions; and (b) s, to mark the literals the truth value of which is derived from the strict rules of $T_v(C)$.

Overall, for a rule r_i, such that the literals in its body are logical consequences of the local strict rules in $T_v(C)$, *Priorities* assigns $\{s\}$ as its Supportive Set ($1(\alpha)$). $\{w\}$ is assigned as the Supportive Set of a rule that contains a foreign literal in its body, which is not a logical consequence of $T_v(C)$ ($1(\beta)$). For the rest of the rules, *Priorities* computes their Supportive Set as the union of foreign literals contained in their body with the Supportive Sets of the local literals in their body ($1(\gamma)$). The priority relation between two applicable rules with contradictory conclusions is computed based on the *Stronger* function, which takes into account the preference order of their context (2). For a literal p_i, which is logical consequence of $T_v(C)$, *Priorities* assigns the Supportive Set of the *strongest* supportive rule with head p_i as the Supportive Set of p_i, using the *Stronger* function ($3(\alpha)$). For the rest of the literals, $\{w\}$ is assigned as their Supportive Set ($3(\beta)$).

In the followings we denote the set of strict rules with head p_i in $T_v(C)$ as $R^s[p_i]$, and the set of defeasible rules with head p_i in $T_v(C)$ as $R^d[p_i]$.

1. *If* $Pr(i+1) = S_{r_i}$ *then* $r_i \in T_v(C)$ *and either*
   (α) $S_{r_i} = \{s\}$ *and* $r_i \in R^s[p_i]$ *and*
   $\forall a_i \in body(r_i)$, $S_{a_i} = \{s\} \in Pr(1...i)$ *or*
   (β) $S_{r_i} = \{w\}$, *and* $r_i \in R^d[p_i]$ *and*
   $\exists a_j \in body(r_i)$: $a_j \notin V_i$, $S_{a_j} \in Pr(1...i)$ *and* $w \in S_{a_j}$ *or*
   (γ) (γ_1) $\forall a_i \in body(r_i) \cap V_i$: $S_{a_i} \in Pr(1...i)$ *and*
   (γ_2) $\forall a_j \in body(r_i) - V_i$: $S_{a_j} \in Pr(1...i)$ *and* $w \notin S_{a_j}$ *and*
   (γ_3) $S_{r_i} = (\bigcup_{a'_i} S_{a'_i}) \cup (\bigcup_{a_j} a_j)$, *where*
   a'_i *are the literals in the body of* r_i *s.t.* $a'_i \in V_i$ *and* $S_{a'_i} \neq \{s\}$

2. *If* $Pr(i+1) = r_i > s_i$ *then* $r_i, s_i \in T_v(C)$ *and*
   (α) $head(r_i) = \sim head(s_i)$ *and*
   (β) $S_{r_i}, S_{s_i} \in Pr(1...i)$ *and*
   (γ) $w \notin S_{r_i}$, $w \notin S_{s_i}$, $S_{r_i} \neq \{s\}$, $S_{s_i} \neq \{s\}$ *and*
   (δ) $Stronger(S_{r_i}, S_{s_i}, T_i) = S_{r_i}$

```
3. If $Pr(i+1) = S_{p_i}$ then either
    ($\alpha$) $\exists r_i \in R[p_i]$: $S_{r_i} \in Pr(1...i)$ and $S_{p_i} = S_{r_i}$ and $w \notin S_{r_i}$ and either
        ($\alpha_1$) $r_i \in R^s[p_i]$ and $S_{r_i} = \{s\}$ or
        ($\alpha_2$) ($\alpha_{2.1}$) $\forall s_i \in R[\sim p_i]$:
                    ($\alpha_{2.1.1}$) $S_{s_i} \in Pr(1...i)$ and
                    ($\alpha_{2.1.2}$) $S_{s_i} \neq \{s\}$ and
                    ($\alpha_{2.1.3}$) $w \in S_{s_i}$ or $Stronger(S_{r_i}, S_{s_i}, T_i) = S_{r_i}$ and
            ($\alpha_{2.2}$) $\forall t_i \in R[p_i]$:
                    ($\alpha_{2.2.1}$) $S_{t_i} \in Pr(1...i)$ and
                    ($\alpha_{2.2.2}$) $S_{t_i} \neq \{s\}$ and
                    ($\alpha_{2.2.3}$) $w \in S_{t_i}$ or $Stronger(S_{r_i}, S_{t_i}, T_i) \neq S_{t_i}$ or
    ($\beta$) $S_{p_i} = \{w\}$ and either
        ($\beta_1$) $\forall r_i \in R[p_i]$: $S_{r_i} \in Pr(1...i)$ and $w \in S_{r_i}$ or
        ($\beta_2$) $\exists s_i \in R[\sim p_i]$:
            ($\beta_{2.1}$) $S_{s_i} \in Pr(1...i)$ and $w \notin S_{s_i}$ and $S_{s_i} \neq \{s\}$ and
            ($\beta_{2.2}$) $\forall r_i \in R[p_i]$: $S_{r_i} \in Pr(1...i)$ and $S_{r_i} \neq \{s\}$ and either
                    ($\beta_{2.2.1}$) $w \in S_{r_i}$ or
                    ($\beta_{2.2.2}$) $w \notin S_{r_i}$ and $Stronger(S_{r_i}, S_{s_i}, T_i) \neq S_{r_i}$ or
        ($\beta_3$) $S_{\sim p_i} \in Pr(1...i)$ and $S_{\sim p_i} = \{s\}$
```

$Pr(n)$ will contain the Supportive Sets of all rules and literals in $T_v(C)$, and the required priority relations between all conflicting rules in $T_v(C)$. It is easy to verify that the process is deterministic in the sense that for a given MCS C, the same set of elements is always contained in $Pr(n)$ regardless of the order of contexts C_i in C, and the orders of the rules in each context C_i in C.

6. Alternative Strategies for Conflict Resolution

The algorithm described in Section 5 (*P2P_DR*) resolves conflicts that arise when mutually inconsistent information is imported from different contexts based on the ranks of these contexts in the preference ordering of the context that imports this information. This section describes three 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 imported knowledge. The intuition behind these strategies is that imported knowledge should be evaluated not only based on its “source” - the context that the knowledge is imported by - but also on the way that the “source” acquired this knowledge. The following sections discuss the features of the three strategies, explain their main differences through examples, and describe their implementation in three alternative versions of *P2P_DR*.

6.1. Strict-Weak Answers

The distinct feature of *Strict-Weak Answers*, compared to the strategy implemented by *P2P_DR* (which for the rest of the paper will be referred to as *Single Answers*), 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. The intuition behind this is that the case that the imported knowledge is part of the local context knowledge of the “source” should be treated differently (preferred) to the case that the “source” may have used knowledge imported from third parties to derive this knowledge. For example, in the case of the system in Example 1 (Figure 1), the knowledge that the mobile phone imports from the laptop and the localization service is part of their local knowledge bases, and hence should be preferred to the knowledge that is imported from the classroom manager, which is derived based on information that the classroom manager imports from an external source.

6.1.1. Distributed Query Evaluation

The version of the algorithm that implements this strategy, $P2P\_DR_{SWA}$, uses two types of answers for the literals with a positive truth value:

– a *strict* answer indicates that the truth value is derived from the strict rules of the queried context;
– a *weak* answer indicates that the truth value is derived from the combination of local and mapping rules of the queried context

For *undefined* answers, there is no need to define two different types. Since we have assumed that there are no loops in the local context theories, an *undefined* answer cannot be derived using only the local strict rules of a context. $P2P\_DR_{SWA}$ follows the four main steps of $P2P\_DR$, but with the following modifications:

– For a literal p_i, Ans_{p_i} can take one of the following four values:
   1. $str(true)$, indicating a strictly derived positive truth value
   2. $weak(true)$ indicating a non-strictly derived positive truth value
   3. $undefined$
   4. $false$
– An element of a Supportive/Blocking Set is actually a signed literal; the sign of the literal indicates whether the truth value of the literal is derived from the local strict rules of the queried context (e.g. $+p_i$), or from the combination of the strict rules and the local defeasible and mapping rules of the context ($-p_i$).
– The strength of an element in a Supportive/Blocking Set is determined primarily by the type of answer (*strict* answers are considered stronger than *weak* ones), and secondly by the rank of the queried context in the preference order of the querying context.

The pseudocode of $P2P\_DR_{SWA}$ is derived from $P2P\_DR$ as follows:

– Lines 2-4 are replaced by:
```
if $localAns_{p_i} = true$ then
    $Ans_{p_i} \leftarrow str(true)$, $SS_{p_i} \leftarrow \emptyset$, $BS_{p_i} \leftarrow \emptyset$
    terminate
```
– Lines 15-16 are replaced by:
```
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
    $Ans_{p_i} \leftarrow weak(true)$
```

local_alg remains unchanged, while *Support* is modified as follows:

– Lines 8-10 are replaced by:
```
    if $b_t \in Hist_{p_i}$ then
        $cycle(r_i) \leftarrow true$
        $BS_{r_i} \leftarrow BS_{r_i} \cup \{-d_t\}$
```
– Lines 18-19 are replaced by:
```
    if $b_t \notin V_i$ then
        $BS_{r_i} \leftarrow BS_{r_i} \cup \{-b_t\}$
```
– Lines 23-25 are replaced by:
```
    if $b_t \notin V_i$ and $Ans_{b_t} = str(true)$ then
        $BS_{r_i} \leftarrow BS_{r_i} \cup \{+b_t\}$
        $SS_{r_i} \leftarrow SS_{r_i} \cup \{+b_t\}$
    else if $b_t \notin V_i$ and $Ans_{b_t} = weak(true)$ then
        $BS_{r_i} \leftarrow BS_{r_i} \cup \{-b_t\}$
        $SS_{r_i} \leftarrow SS_{r_i} \cup \{-b_t\}$
```

Finally, the *Stronger* function is also modified as follows:

Parameters
A, B: sets of signed literals of the form $+p_i/-p_i$
T_i: a preference order

Stronger(A, B, T_i)
```
1: if $\exists b_l$: $-b_l \in B$ and $\forall \pm a_k \in A$ either $+a_k \in A$ or $C_k$ has lower rank than $C_l$ in $T_i$ then
2:     $Stronger = A$
3: else if $\exists a_k$: $-a_k \in A$ and $\forall \pm b_l \in B$ either $+b_l \in B$ or $C_l$ has lower rank than $C_k$ in $T_i$ then
4:     $Stronger = B$
5: else if $\forall \pm a_k \in A, \pm b_l \in B$: $+a_k \in A$ and $+b_l \in B$ then
6:     if $\exists + b_l \in B$: $\forall + a_k \in A$, $C_k$ has lower rank than $C_l$ in $T_i$ then
7:         $Stronger = A$
8:     else if $\exists + a_k \in A$: $\forall + b_l \in B$, $C_l$ has lower rank than $C_k$ in $T_i$ then
9:         $Stronger = B$
10: else
11:     $Stronger = None$
```

Example 2. Consider that a query about x_1 is issued to C_i in the MCS depicted in Figure 2. An interesting feature of this system is that a_3 and a_4, which constitute the premises of r^m_{13}, are strict consequences of the local theories of C_3 and C_4 respectively, while a_2, which is the only premise of the rule that is in conflict with r^m_{13} (r^m_{12}) is not a strict consequence of the local theory of C_2.

P2P_DR, the algorithm that implements the *Single Answers* strategy, receives positive answers (*true*) for a_2, a_3 and a_4 (from the instances of the algorithm called by C_2, C_3 and C_4 respectively), and resolves the conflict that arises for literal a_1 by comparing its Supportive Set, $SS_{a_1} = SS_{r_{12}} = \{a_2\}$, with the Blocking Set of $\neg a_1$, $BS_{\neg a_1} = BS_{r_{13}} = \{a_3, a_4\}$. Assuming that the preference order defined by C_1 is $T_1 = [C_4, C_2, C_6, C_3, C_5]$, *P2P_DR* determines that SS_{a_1}

$\underline{C_1}$	$\underline{C_2}$	$\underline{C_3}$
$r^l_{11} : a_1 \rightarrow x_1$	$r^l_{21} : c_2 \rightarrow a_2$	$r^l_{31} :\rightarrow a_3$
$r^m_{12} : a_2 \Rightarrow a_1$	$r^l_{22} : b_2 \rightarrow a_2$	
$r^m_{13} : a_3, a_4 \Rightarrow \neg a_1$	$r^m_{23} : b_5 \Rightarrow b_2$	
	$r^m_{24} : b_6 \Rightarrow b_2$	
$\underline{C_4}$	$\underline{C_5}$	$\underline{C_6}$
$r^l_{41} :\rightarrow a_4$	$r^l_{51} :\rightarrow b_5$	$r^l_{61} :\rightarrow b_6$

Fig. 2. A MCS of Six Context Theories (example 6)

is *stronger* than $BS_{\neg a_1}$ (as C_2 precedes C_3 in T_1) and returns a positive answer for a_1 and eventually for x_1 as well.

On the other hand, $P2P\_DR^{SWA}$ proceeds as follows: For a_3 and a_4, it returns $Ans_{a_3} = str(true)$ and $Ans_{a_3} = str(true)$, as both are strict local conclusions of C_3 and C_4, respectively. For a_2, it returns $Ans_{a_3} = weak(true)$, as for the evaluation of this answer it uses both local and mapping rules of C_2. Hence, the Supportive Sets of rules r_{12} and r_{13} are respectively: $SS_{r^m_{12}} = \{-a_2\}$, and $SS^m_{r_{13}} = \{+a_3, +a_4\}$. Hence, $BS_{a_1} = \{-a_2\}$ and $SS_{\neg a_1} = \{+a_3, +a_4\}$, and $SS_{\neg a_1}$ is *stronger* than BS_{a_1}. Eventually, $P2P\_DR^{SWA}$ returns negative truth values for a_1 and, hence for x_1 as well.

Example 1 (continued). In the MCS of example 1, $P2P\_DR$ receives positive answers (*true*) for $classtime_2$, $location\_RA201_3$ and $\neg class\_activity_4$ (from the instances of the algorithm called by C_2, C_3 and C_4 respectively), and resolves the conflict that arises for $\neg lecture_1$ by comparing its Supportive Set, $SS_{\neg lecture_1} = SS_{r^m_{16}} = \{\neg class\_activity_4\}$, with the Blocking Set of $lecture_1$, $BS_{lecture_1} = BS_{r^m_{15}} = \{classtime_2, location\_RA201_3\}$. Using $T_1 = [C_4, C_3, C_2, C_5]$, $P2P\_DR$ determines that $SS_{\neg lecture_1}$ is *stronger* than $BS_{lecture_1}$ (as C_4 precedes both C_2 and C_3 in T_1) and returns a positive answer for $\neg lecture_1$ and eventually for $ring_1$ as well.

On the other hand, the version of the algorithm that implements *Strict-Weak Answers*, $P2P\_DR^{SWA}$, returns $Ans_{classtime_2} = str(true)$ and $Ans_{location\_RA201_3} = str(true)$, as both are strict local conclusions of C_2 and C_3 respectively. For $\neg class\_activity_4$, it returns $Ans_{\neg class\_activity_4} = weak(true)$, as for the evaluation of this answer it uses both local and mapping rules of C_4. Hence, the Supportive Sets of rules r^m_{15} and r^m_{16} are respectively: $SS_{r^m_{12}} = \{+classtime_2, +location\_RA201_3\}$, and $SS^m_{r_{13}} = \{-\neg class\_activity_4\}$, and $P2P\_DR^{SWA}$ evaluates a negative truth value for $\neg lecture_1$ and, eventually returns a negative truth value for $ring_1$ as well.

6.1.2. Complexity Analysis

The implementation of the *Strict-Weak Answers* strategy does not require drastic modifications to the algorithms that implement the *Single Answers* strategy. Specifically, *local_alg* remains as it is, while the modifications of *Support*, *Stronger* and *P2P_DR* do not affect the overall complexity of the algorithms. Therefore, Proposition 2 holds as it is for the versions of the algorithms that implement the *Strict-Weak Answers* strategy as well.

Similarly with the case of $P2P\_DR$, which implements the *Single Answers* strategy, $P2P\_DR_{SWA}$ can also be optimized using structures that retain the answers for incoming and outgoing queries. In fact, $P2P\_DR_{SWA}$ uses structures of exactly the same form with INC_Q and OUT_Q. The only difference is that for a record for $(p_j, Hist_{p_j})$, the field that corresponds to Ans_{p_j} is filled with one of the values: $str(true)$, $weak(true)$, $undefined$ and $false$.

Using these optimizations, the number of algorithm calls and messages is reduced in the same way with the *Single Answers* strategy, to $O(n \times \sum P(n,k))$ and $O(2 \times n \times \sum P(n,k))$ respectively for the general case, and $O(c \times n)$ and $O(2 \times c \times n)$ respectively for acyclic MCS, where c stands for the total number of contexts and n for the total number of literals in the system.

6.2. Propagating Mapping Sets

The third strategy, *Propagating Mapping Sets*, goes one step further compared to *Strict-Weak Answers*. Similarly with the second strategy, the evaluation of imported knowledge is based on how the "source" of this knowledge infers this knowledge. In *Strict-Weak Answers*, we only care about whether the imported knowledge is inferred from the strict local knowledge of the "source" or it is derived from both its local theory and its mappings (and therefore from the knowledge of other contexts). *Propagating Mapping Sets* further requires 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. In the MCS of example 1 (Figure 1), following this strategy, the classroom manager does not only notify the mobile phone that there is no class activity in the classroom, but also informs it that to reach this conclusion it had to import knowledge from the person detection service. Hence, the evaluation of this knowledge will take into account the preference that the mobile phone has in both the classroom manager and the person detection service.

6.2.1. Distributed Query Evaluation

The only modifications of $P2P\_DR$ that are required to implement this strategy are two:

– the Supportive Set and the Blocking Set of the queried literal are always returned along with the computed truth value - in contrast with $P2P\_DR$, which in case that a query is posed by some other context, the queried context returns only the truth value of the literal it is queried about.
– The Supportive (Blocking) Set of a rule is the union of the foreign literals in the body of the rule and the Supportive (Blocking) Sets of all literals in the body of the rule - in contrast with $P2P\_DR$, where the Supportive (Blocking) Sets of the foreign literals are not taken into account.

The pseudocode of $P2P\_DR_{PS}$, the version of the distributed algorithm that implements *Propagating Mapping Sets*, is derived from $P2P\_DR$ as follows: Lines 15-24 are replaced by:

```
**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**
```

```
    $Ans_{p_i} \leftarrow true$
  else if $sup_{\sim p_i} = true$ and $Stronger(BS_{p_i}, SS_{\sim p_i}, T_i) \neq BS_{p_i}$ then
    $Ans_{p_i} \leftarrow false$, $SS_{p_i} = \emptyset$, $BS_{p_i} = \emptyset$
  else
    $Ans_{p_i} \leftarrow undefined$
```

The pseudocodes of *local_alg* and *Stronger* remain unchanged, while *Support* is modified as follows:

– Lines 8-10 are replaced by:

```
    if $b_t \in Hist_{p_i}$ then
      $cycle(r_i) \leftarrow true$
      $BS_{r_i} \leftarrow BS_{r_i} \cup (\bigcup\{d_t\})$ {$d_t$ are the foreign literals of $C_i$ added in $Hist_{p_i}$
      after $b_t$ including $b_t$ in case $b_t \notin V_i$}
```

– Lines 18-19 are replaced by:

```
    if $b_t \notin V_i$ then
      $BS_{r_i} \leftarrow BS_{r_i} \cup BS_{b_t} \cup \{b_t\}$
```

– Lines 23-25 are replaced by:

```
    if $b_t \notin V_i$ then
      $BS_{r_i} \leftarrow BS_{r_i} \cup BS_{b_t} \cup \{b_t\}$
      $SS_{r_i} \leftarrow SS_{r_i} \cup SS_{b_t} \cup \{b_t\}$
```

Example 2 (continued). In the MCS depicted in Figure 2, $P2P\_DR_{PS}$ returns positive (*true*) answers for a_3, a_4, b_5 and b_6 along with empty Supportive and Blocking Sets, as all of them are conclusions of C_3, C_4, C_5 and C_6 respectively. Called by C_2, *Support* computes $SS_{r^m_{23}} = \{b_5\}$ and $SS_{r^m_{24}} = \{b_6\}$. Assuming that the preference ordering defined by C_2 is $T_2 = [C_4, C_3, C_5, C_2, C_6]$, *Support* determines that $SS_{r^m_{23}}$ is stronger than $SS_{r^m_{24}}$, and $P2P\_DR_{PS}$ returns *true* as an answer for b_2 and a_2 and $SS_{a_2} = SS_{b_2} = \{b_5\}$. For r^m_{12} and r^m_{13}, *Support* respectively computes $SS_{r^m_{12}} = \{a_2, b_5\}$ and $SS_{r^m_{13}} = \{a_3, a_4\}$. According to $T_1 = [C_4, C_2, C_6, C_3, C_5]$, C_5 does not precede neither C_3 nor C_4 in T_1. Therefore, $SS_{\neg a_1} = SS_{r^m_{13}}$ is computed to be *stronger* than $BS_{a_1} = SS_{r^m_{12}}$, and $P2P\_DR_{PS}$ returns *false* as answer for a_1, and eventually $Ans_{x_1} = false$.

Example 1 (continued) In the MCS of Example 1 (Figure 1), $P2P\_DR_{PS}$ returns positive (*true*) answers for $classtime_2$, $location\_RA201_3$, $projector(off)_4$ and $detected(1)_5$ along with empty Supportive and Blocking Sets, as all of them are included in the strict local knowledge of C_2 (laptop), C_3 (localization service), C_4 (classroom manager) and C_5 (person detection service) respectively. For $\neg class\_activity_4$, it returns a positive (*true*) answer along with its Supportive Set $SS_{\neg class\_activity_4} = \{detected(1)_5\}$. For the two conflicting rules, r^m_{15} and r^m_{16}, *Support* respectively computes $SS_{r^m_{15}} = \{classtime_2, location\_RA201_3\}$ and $SS_{r^m_{16}} = \{\neg class\_activity_4, detected(1)_5\}$. According to $T_1 = [C_4, C_3, C_2, C_5]$, C_5 does not precede neither C_3 nor C_4 in T_1. Therefore, $SS_{lecture_1} = SS_{r^m_{15}}$ is computed to be *stronger* than $BS_{\neg lecture_1} = SS_{r^m_{16}}$, and $P2P\_DR_{PS}$ returns *false* as answer for $\neg lecture_1$, and eventually $Ans_{ring_1} = false$.

6.2.2. Complexity Analysis

The implementation of the *Propagating Mapping Sets* strategy requires each context to return the Supportive and Blocking Sets of the queried literal, along with the answer that indicates its truth value. This affects the overall complexity of distributed query evaluation as follows: The overall complexity of the algorithms remains the same with the case of *Single Answers*. However, the *worst case* is not the same in the two cases. The worst case in *Single Answers* is when all rules of C_i (the context that evaluates the query) contain either p_i or $\sim p_i$ in their heads and all literals defined in the system in their bodies. In *Propagating Mapping Sets*, the worst case is when all rules of C_i contain either p_i or $\sim p_i$ in their heads, and the Supportive / Blocking Sets of all rules contain all literals defined in the system, which means that the applicability of each rule in C_i depends on the truth values of all literals defined in the system. Obviously, the worst case described for *Single Answer* is a subcase of the case that we describe here. This result is described in the Proposition below.

Proposition 6. The number of operations imposed by one call of $P2P\_DR$ for the evaluation of a query for literal p_i is in the worst case that all rules of C_i contain either p_i or $\sim p_i$ in their head, and the applicability of each rule in C_i depends on the truth values of all literals defined in the system, proportional to the number of rules in C_i, and to the total number of literals in the system.

Similarly with the case of $P2P\_DR$, which implements the *Single Answers* strategy, $P2P\_DR_{PS}$ can also be optimized using structures that retain the answers for incoming and outgoing queries. In this case both INC_Q and OUT_Q contain records of the form: $rec(p_i, Hist_{p_i}) : (Ans_{p_i}, BS_{p_i}, SS_{p_i})$ for each query about local/foreign p_i with history $Hist_{p_i}$ that has already been evaluated.

Using these optimizations, the number of algorithm calls and messages is reduced in the same way with the *Single Answers* strategy, to $O(n \times \sum P(n,k))$ and $O(2 \times n \times \sum P(n,k))$ respectively for the general case, and $O(c \times n)$ and $O(2 \times c \times n)$ respectively for acyclic MCS, where c stands for the total number of contexts and n for the total number of literals in the system.

6.3. Complex Mapping Sets

The main feature of *Propagating Mapping Sets* is that a context that imports knowledge from another context, requires the "source" to return information about the identity of all other contexts that are involved in the derivation of the imported knowledge. Specifically, the "source" returns a set of literals that corresponds to the *most preferred* reasoning chain that leads to the inferred knowledge. However, preference is subjective. The *most preferred* between two or more different reasoning chains may vary according to the viewpoint of two different contexts. The fourth strategy, *Complex Mapping Sets*, 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. In the MCS used in Example 1, for each piece of knowledge that is exchanged between the ambient agents, there is only one reasoning chain that leads to its inference. Suppose, however, that there are two different services that provide knowledge about the presence of people in the classroom, C_5 and C_6, and that both detect one person in the classroom. Using the strategy described in the previous section

(*Propagating Mapping Sets*), the classroom manager will use its local preference ordering to determine which of them is *preferred* (e.g. more trustworthy), and will return the identity of that service (e.g. C_5) to the mobile phone. In this case, the mobile phone cannot be aware that C_6 also provided the same knowledge, and will evaluate the knowledge it imports from the classroom manager based on its preference in the classroom manager and C_5. Following *Complex Mapping Sets*, the classroom manager will inform the mobile phone that there are two different ways to infer that there is no activity in the classroom; one that involves knowledge derived from C_5, and another one that involves the local knowledge of C_6. In this case, the mobile phone will separately evaluate the two different reasoning chains, using its own preference in the two person detection services. If at least one of these chains is preferred to the reasoning chains that lead to contradictory conclusions, the mobile phone will be able to use this knowledge (that there is no class activity) to derive further conclusions. Obviously, 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.

6.3.1. Distributed Query Evaluation

To support the features of *Complex Mapping Sets*, *P2P_DR* is modified as follows:

– the Supportive Set and the Blocking Set of the queried literal are always returned along with the computed truth value.
– The Supportive Set (Blocking Set) of a literal is actually the set of the Supportive Sets (Blocking Sets) of all rules that can be applied to support this literal.
– The Supportive Set (Blocking Set) of a rule is the *Union Product* (Definition 1) of the Supportive Sets (Blocking Sets) of all literals in the body of the rule.

To describe the algorithms that implement *Complex Mapping Sets*, we use the notion of *Union Product*.

Definition 1. Let A, B be two sets, the elements of which are sets of literals. Their Union Product is defined as:

$$A \otimes B = \{a_i \cup b_j | a_i \in A, b_j \in B\}$$

The Union Product of n sets $A_1, A_2, ..., A_n$ is defined as follows:

$$\bigotimes_i A_i = (...((A_1 \otimes A_2) \otimes A_3)... \otimes A_n)$$

P2P_DR$_{CS}$ is derived from *P2P_DR* by replacing lines 15-24 with:

```
**if** $sup_{p_i} = true$ and
($unb_{\sim p_i} = false$ or $\exists A \in BS_{p_i}$: $\forall B \in SS_{\sim p_i}$ $Stronger(A, B, T_i) = A$) **then**
    $Ans_{p_i} \leftarrow true$
**else if** $sup_{\sim p_i} = true$ and $\exists B \in SS_{\sim p_i}$: $\forall A \in BS_{p_i}$ $Stronger(A, B, T_i) \neq A$
**then**
    $Ans_{p_i} \leftarrow false$, $SS_{p_i} = \emptyset$, $BS_{p_i} = \emptyset$
**else**
    $Ans_{p_i} \leftarrow undefined$
```

The pseudocodes of *local_alg* and *Stronger* remain unchanged, while *Support* is modified as follows:

```
Support($p_i, Hist_{p_i}, T_i, sup_{p_i}, unb_{p_i}, SS_{p_i}, BS_{p_i}$)
1: $sup_{p_i} \leftarrow false$
2: $unb_{p_i} \leftarrow false$
3: for all $r_i \in R[p_i]$ do
4:    $cycle(r_i) \leftarrow false$
5:    $SS_{r_i} \leftarrow \emptyset$
6:    $BS_{r_i} \leftarrow \emptyset$
7:    for all $b_t \in body(r_i)$ do
8:       if $b_t \in Hist_{p_i}$ then
9:          $cycle(r_i) \leftarrow true$
10:         $BS_{r_i} \leftarrow BS_{r_i} \otimes (\bigcup\{d_t\})$ {$d_t$ are the foreign literals of $C_i$ added in
            $Hist_{p_i}$ after $b_t$ including $b_t$ in case $b_t \notin V_i$}
11:      else
12:         $Hist_{b_t} \leftarrow Hist_{p_i} \cup \{b_t\}$
13:         call $P2P\_DR(b_t, C_i, C_t, Hist_{b_t}, T_t, SS_{b_t}, BS_{b_t}, Ans_{b_t})$
14:         if $Ans_{b_t} = false$ then
15:            stop and check the next rule
16:         else if $Ans_{b_t} = undefined$ or $cycle(r_i) = true$ then
17:            $cycle(r_i) \leftarrow true$
18:            if $b_t \notin V_i$ then
19:               $BS_{r_i} \leftarrow BS_{r_i} \otimes (BS_{b_t} \otimes \{b_t\})$
20:            else
21:               $BS_{r_i} \leftarrow BS_{r_i} \otimes BS_{b_t}$
22:         else
23:            if $b_t \notin V_i$ then
24:               $BS_{r_i} \leftarrow BS_{r_i} \otimes (BS_{b_t} \otimes \{b_t\})$
25:               $SS_{r_i} \leftarrow SS_{r_i} \otimes (SS_{b_t} \otimes \{b_t\})$
26:            else
27:               $BS_{r_i} \leftarrow BS_{r_i} \otimes BS_{b_t}$
28:               $SS_{r_i} \leftarrow SS_{r_i} \otimes SS_{b_t}$
29:   $BS_{p_i} \leftarrow BS_{p_i} \cup BS_{r_i}$
30:   $unb_{p_i} \leftarrow true$
31:   if $cycle(r_i) = false$ then
32:      $SS_{p_i} \leftarrow SS_{p_i} \cup SS_{r_i}$
33:      $sup_{p_i} \leftarrow true$
```

Example 2 (continued). In the MCS depicted in Figure 2, $P2P\_DR_{CS}$ returns positive (*true*) answers for a_3, a_4, b_5 and b_6 along with empty Supportive and Blocking Sets, as all of them are local conclusions of C_3, C_4, C_5 and C_6 respectively. Called by C_2, *Support* computes $SS_{r^m_{23}} = \{\{b_5\}\}$ and $SS_{r^m_{24}} = \{\{b_6\}\}$, and $P2P\_DR_{CS}$ returns *true* as an answer for b_2 and a_2 and $SS_{a_2} = SS_{b_2} = \{\{b_5\}, \{b_6\}\}$. For r^m_{12} and r^m_{13}, *Support* respectively computes $SS_{r^m_{12}} = \{\{a_2, b_5\}, \{a_2, b_6\}\}$ and $SS_{r^m_{13}} = \{\{a_3, a_4\}\}$, and $SS_{a_1} = SS_{r^m_{12}}$, $BS_{\neg a_1} = SS_{r^m_{13}}$. According to $T_1 = [C_4, C_2, C_6, C_3, C_5]$, C_6 and C_2 both precede C_3 in T_1, and $A = \{a_2, b_6\} \in SS_{a_1}$ is computed to be *stronger* than $B = \{a_3, a_4\}$, which is the only set in $BS_{\neg a_1}$, and therefore $P2P\_DR_{CS}$ returns *true* as answer for a_1, and eventually $Ans_{x_1} = true$.

Example 1 (continued). In the system described in Example 1 (Figure 1), suppose that there is an additional person detection service, C_6, and its local knowledge is encoded in rule r^l_{61}, which states that it has detected one person it the classroom:

$$r^l_{61} :\rightarrow persons(1)_6$$

Suppose also that C_4 (the classroom manager) uses an additional mapping rule, r^m_{43}, which states that if the projector is off, and it receives information from C_6 that there is only one person in the classroom, then there is no class activity.

$$r^m_{43} : projector(off)_4, persons(1)_6 \Rightarrow \neg class\_activity_4$$

Suppose also that the preference orderings of C_1 and C_4 are, respectively, $T_1 = [C_4, C_3, C_6, C_2, C_5]$ and $T_4 = [C_4, C_3, C_5, C_2, C_6]$. $P2P\_DR_{PS}$, the algorithm that implements *Propagating Mapping Sets*, will compute $Ans_{\neg class\_activity_4} = true$ and $SS_{\neg class\_activity_4} = \{detected(1)_5\}$, since C_5 precedes C_6 in T_4. *Support* will compute $SS_{r^m_{15}} = \{classtime_2, location\_RA201_3\}$ and $SS_{r^m_{16}} = \{\neg class\_activity_4, detected(1)_5\}$. Since C_5 does not precede neither C_3 nor C_4 in T_1, $SS_{lecture_1} = SS_{r^m_{15}}$ will be computed to be *stronger* than $BS_{\neg lecture_1} = SS_{r^m_{16}}$, and $P2P\_DR_{PS}$ will return *false* as answer for $\neg lecture_1$, and eventually $Ans_{ring_1} = false$.

On the other hand, $P2P\_DR_{CS}$ will also return $Ans_{\neg class\_activity_4} = true$, but the Supportive Set of $\neg class\_activity_4$ is in this case $SS_{\neg class\_activity_4} = \{\{detected(1)_5\}, \{persons(1)_6\}\}$. For the two conflicting rules, r^m_{15} and r^m_{16}, *Support* respectively computes $SS_{r^m_{15}} = \{\{classtime_2, location\_RA201_3\}\}$ and $SS_{r^m_{16}} = \{\{\neg class\_activity_4, detected(1)_5\}, \{\neg class\_activity_4, persons(1)_6\}\}$. According to T_1, C_6 and C_4 both precede C_2 in T_1, and $A = \{\neg class\_activity_4, persons(1)_6\} \in SS_{\neg lecture_1}$ is computed to be *stronger* than $B = \{classtime_2, location\_RA201_3\}$, which is the only set in $BS_{lecture_1}$. Eventually, $P2P\_DR_{CS}$ returns *true* as answer for $\neg lecture_1$, and $Ans_{ring_1} = true$.

6.3.2. Complexity Analysis

The main difference of *Complex Mapping Sets* with the first three strategies is that the Supportive / Blocking Sets used in this case are actually sets of sets of literals, with each different set representing a different way to prove a literal. As a result, each Supportive / Blocking Set may contain a number of sets, which is in the worst case equal to the total number of different combinations of the literals defined in the system. As a result, the complexity of comparing two Supportive / Blocking Sets is in this case $O(n^n)$, where n is the total number of literals defined in the system, making the overall complexity of the algorithms exponential to the size of the knowledge base.

Proposition 7. The number of operations imposed by one call of $P2P\_DR$ for the evaluation of a query for literal p_i is in the worst case that all rules of C_i contain either p_i or $\sim p_i$ in their head, and each different combination of system literals can be used to derive either p_i or $\sim p_i$, $O(n^n)$, where n is the total number of literals defined in the system.

$P2P\_DR_{CS}$ can also be optimized using structures of the same form with those used by $P2P\_DR_{PS}$, which implements *Propagating Mapping Sets*. Each record of INC_Q or OUR_Q is of the form: $rec(p_i, Hist_{p_i}) : (Ans_{p_i}, BS_{p_i}, SS_{p_i})$, and the only difference with the respective structures used by $P2P\_DR_{PS}$ is in the form of BS_{p_i} and SS_{p_i}.

Using these optimizations, the number of algorithm calls and messages is reduced in the same way with the *Single Answers* strategy, to $O(n \times \sum P(n,k))$ and $O(2 \times n \times \sum P(n,k))$ respectively for the general case, and $O(c \times n)$ and $O(2 \times c \times n)$ respectively for acyclic MCS, where c stands for the total number of contexts and n for the total number of literals in the system.

7. Prototypical Implementation and Experimental Evaluation

This section describes an implementation of the four strategies for conflict resolution in a simulated peer-to-peer environment, and the results of an experimental evaluation of the four strategies in terms of computational complexity.

7.1. Simulation Environment

In order to evaluate the four strategies, we implemented the respective versions of $P2P\_DR$, and a P2P system simulating the proposed Multi-Context framework in Java. The main reasons for choosing this particular programming language are

1. Java contains several data structures that can be used easily and efficiently.
2. It is a *"write-once, use many"* language, thus giving us the opportunity to use the peer-to-peer system virtually anywhere a virtual machine can be installed, from personal computers to mobile phones. This adds an extra advantage when creating applications that involve multiple types of devices.

For the network library as well as the peer-to-peer communication library, we used a custom-built library based on the java.network packages. Libraries such as *JXTA* (Gong, 2001) would be inefficient due to the complexity in configuring such a simple ad-hoc peer-to-peer network. The message exchanging protocol in our custom library is also simple and straightforward. However, one can use any other peer communication libraries, as the system uses an abstract network manager interface.

The system is composed of 5 packages: *agencies*, *logic*, *knowledge*, *network*, *peerlib*. The *agencies* package contains the classes that implement the text file parsers as well as those that implement the four versions of $P2P\_DR$. The *logic* package contains the classes the represent (in memory) the literals and the rules. The *knowledge* package includes the *KnowledgeBase* class, which stores the local and mapping rules, the preference ordering and any other required information, and some cache classes. The *network* package includes the mechanism that associates a new socket connection with a new thread, whereas the *peerlib* contains the higher-level classes that operate the communication between two peers.

Figure 3 depicts the architecture, organized in a protocol stack manner. The main class that operates the peer instance is called *Node*. Its functionality includes parsing the preference ordering and theory files, as well as the initialization

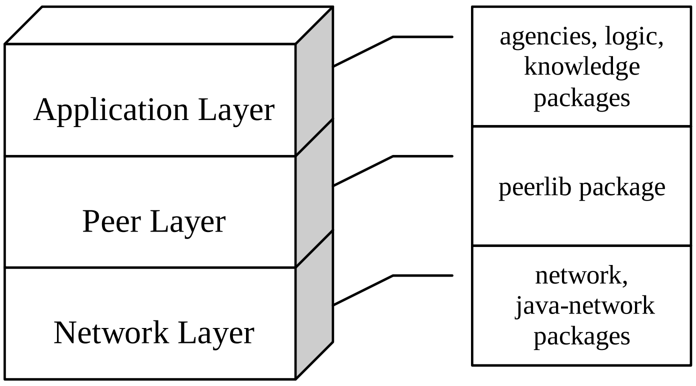
Application Layer
Peer Layer
Network Layer
agencies, logic, knowledge packages
peerlib package
network, java-network packages

Fig. 3. System Layered Architecture.

of the network libraries and knowledge base. When initialization is complete, it waits for pending queries. Finally, another class named *Client* can be used to connect to a *Node* specified by IP address, so that one can manually make specific queries to that specific peer instance.

7.2. Experimental Evaluation

The goal of the experiments was to compare the four different strategies in terms of actual computational time spent by a system peer to evaluate the answer to a single query, and to test their scalability. Below we present the test theories that we used and the setup of the experiments, and discuss the results of the evaluation of the four strategies using systems with various peer populations.

7.2.1. Setup of the Experiments

Using a tool that we built for the needs of the experiments, we created theories that correspond to the case that the evaluation of a single query requires the evaluation of the truth values of all literals from all system nodes. For sake of simplicity, we did not include the case of loops in the global knowledge base; hence, for each literal the returned answer was either *true* or *false* The test theories have the following form:

$r_1^m : a_2, a_3, ..., a_n \Rightarrow a_0$
$r_2^m : a_1, a_3, ..., a_n \Rightarrow a_0$
...
$r_{n/2}^m : a_1, ..., a_{n/2-1}, a_{n/2+1}, ..., a_n \Rightarrow a_0$
$r_{n/2+1}^m : a_1, ..., a_{n/2}, a_{n/2+2}, ..., a_n \Rightarrow \neg a_0$
...
$r_n^m : a_1, a_2, ..., a_{n-1} \Rightarrow \neg a_0$

The above mapping rules are defined by C_0 and associate the truth value of its local literal a_0 with the truth values of the literals from n other system peers. Half of them support a_0 as their conclusion, while the remaining rules contradict a_0 ($\neg a_0$ is in their head). In case the answers returned for all foreign literals a_1,

a_2,...,a_n are all *true*, then all mapping rules are applicable and are involved in the computation of the truth value of a_0.

7.2.2. Number of Messages

As it has already been proved, the number of messages that are required for the computation of a single query is the same for all alternative strategies. Specifically, for the case that we describe above, given a query about a_o and using the optimized algorithms for query evaluation, C_0 will make one query for each of the foreign literals that appear in the body of its mapping rules, sending in total *n query messages*, while it will receive one response for each of the query messages. In total *n response messages* will be received regardless of the followed strategy. In the same sense, assuming that all system peers use theories of the same form, each of the peers that receive *query messages* from C_0 has to make (in the worst case) one query for each of the foreign literals that appear in the body of its mapping rules (n query messages) and will receive an equal number of *query responses*. Hence, totally, in the worst case that *P2P_DR* uses all mapping rules from all system peer theories, the total number of messages that need to be exchanged for the evaluation of the query about a_0 are $4n^2$.

7.2.3. Size of Messages

A major difference between the four versions of *P2P_DR* that implement the four alternative strategies for conflict resolution, is in the size of messages exchanged between the system contexts. Specifically, the size of the query messages is the same for all strategies. Each such message will contain the queried literal, the ids of the querying and the queried contexts and a set of literals representing the *history* of the query. However, the size and form of the query responses largely depend on the conflict resolution strategy. Specifically, for the type of experiments that we conducted, a response message for a literal a_i has one of the following forms:

– In the case of *P2P_DR*, which implements the *Single Answers* strategy, the response message will contain only the truth value of a_i; namely, one of the values *true* and *false* (since there are no loops in the global knowledge base).
– In the case of $P2P\_DR_{SWA}$, which implements the *Strict-Weak Answers* strategy, the response message will contain one of the values *str(true)*, *weak(true)* and *false.*
– In the case of $P2P\_DR_{PS}$, which implements the *Propagating Mapping Sets* strategy, the response message will contain the truth value of a_i (either *true* or *false*) and two sets of literals - SS_{a_i} representing the Supportive Set of a_i and BS_{a_i} representing the Blocking Set of a_i. By construction of the test theories, each of the two sets will contain one literal from each of the peers (except C_i) in the system. Therefore, the size of a response message is in this case $2 \times (n-1) + 1$, where n is the total number of peers in the system.
– In the case of $P2P\_DR_{CS}$, which implements the *Complex Mapping Sets* strategy, the response message will contain the truth value of a_i (either *true* or *false*) and two sets of literals - SS_{a_i} representing the Supportive Set of a_i and BS_{a_i} representing the Blocking Set of a_i. By construction of the test theories, each of the two sets will contain $n-1$ different sets of literals, and each of these sets will contain $n-2$ literals. Therefore, the total size of each response message

Table 1. Size of Response Messages for the Four Strategies

SA	**SWA**	**PS**	**CS**
1	1	$2 \times (n-1) + 1$	$2 \times (n-1) \times (n-2) + 1$

Table 2. Processing Time for the Four Strategies

♯ peers (n)	**SA**	**SWA**	**PS**	**CS**
10	78	80	1313	2532
20	469	540	1534	4305
40	2422	3102	3466	207828
60	5719	6390	7188	-
80	10437	10302	15484	-
100	16484	15550	27484	-

will be $2 \times (n-1) \times (n-2) + 1$, where n is the total number of peers in the system.

The results about the size of *response messages* are summarized in Table 1.

7.2.4. Processing Time

In order to exclude the communication overhead from the total time spent by C_0 to evaluate the truth value of a_0, we filled a local cache class with appropriate answers for all the foreign literals. Specifically, for each version of $P2P\_DR$, this class is filled with answers for all foreign literals $(a_1, a_2, ..., a_n)$ as follows:

1. $P2P\_DR$: Positive truth values (*true*) for all literals
2. $P2P\_DR_{SWA}$: Positive strict/weak answers (chosen randomly) for all literals.
3. $P2P\_DR_{PS}$: Positive truth values with Supportive Sets that contain all other foreign literals:
$$SS_{a_i} = \{a_1, a_2, ..., a_{i-1}, a_{a+1}, ..., a_n\}$$
4. $P2P\_DR_{CS}$: Positive truth values with Supportive Sets of the form:
$$SS_{a_i} = \{\{a_2, ..., a_{i-1}, a_{a+1}, ..., a_n\}, \{a_1, a_3, ..., a_{i-1}, a_{a+1}, ..., a_n\},$$
$$..., \{a_1, a_2, ..., a_{i-1}, a_{a+1}, ..., a_{n-1}\}$$

For each version of the algorithm, we conducted six experiments with a variant size of the global knowledge base in terms of total number of literals, which in this case coincides with the total number of system peers: 10, 20, 40, 60, 80, and 100. The test machine was an Intel Celeron M at 1.4 GHz with 512 MB of RAM.

Table 2 shows in milliseconds the processing time for each version of $P2P\_DR$ (*SA* refers to $P2P\_DR$, *SWA* refers to $P2P\_DR_{SWA}$, *PS* stands for $P2P\_DR_{PS}$, which implements *Propagating Mapping Sets*, while *CS* refers to $P2P\_DR_{CS}$. For the case of $P2P\_DR_{CS}$, we were able to measure the computation time only for the cases where $n = 10, 20, 40$; in the other cases the test machine ran out of memory.

As it is obvious from Table 2, the results for the first three strategies are similar; the computation time is proportional to the square of the number of system peers, verifying our expectations from the theoretical results presented in the previous sections. The *Complex Sets* strategy requires much more memory space and computation time (exponential to the number of peers), which make it inapplicable in cases of very dense systems. The results also verify the tradeoff between the computational complexity and the extent of knowledge that each algorithm exploits to evaluate the quality of the imported context information.

8. Discussion

In Sections 5 and 6 of the paper we described four different strategies for resolving conflicts that arise from the interaction of contexts in a Multi-Context System. These conflicts, which we refer to as *global conflicts*, are caused by the mappings that associate the knowledge of different agents, and are resolved based on the preference orderings of the agents. While in the first strategy (*Single Answers*), the evaluation of imported knowledge is based only on the confidence that the agent that imports the knowledge has in the agent that the knowledge is imported by, the other three strategies take also into account the way that the latter agent acquired this knowledge. In the second strategy (*Strict-Weak Answers*), we only care whether the imported knowledge is part of the strict local knowledge of the agent. In the latter two strategies, we also take into account the other agents (contexts) that are involved in the derivation of this knowledge. In Section 7, we discussed the tradeoff between the computational complexity of each strategy and the extent of knowledge that each strategy exploits in order to evaluate imported knowledge. But are these two the only factors that an agent should consider when choosing a particular strategy? And can we enable each agent in a system to follow a different strategy?

8.1. Choosing the best strategy

Computational complexity is certainly an important factor for choosing the *best* strategy. An agent lying on a device with limited computational capabilities should follow one of the first two strategies, so that it can evaluate the imported knowledge and reach a decision in time. On the other hand, agents lying on machines with enhanced capabilities would prefer the fourth strategy, as it takes into account all information that is available in the system to evaluate imported knowledge.

Another important factor, that we have not yet discussed, is *privacy.* In Section 2.4, we stated that one of the assumptions that we have implicitly made is that each agent is willing to disclose and share part of its local knowledge with all other agents. However, the third and fourth strategies (*Propagating Mapping Sets* and *Complex Mapping Sets*) additionally require the agents to disclose the identities of the agents that are involved in the derivation of their knowledge. We argue, that in real Ambient Intelligence environments or similar settings that our methods can be applied (e.g. social networks), disclosing this type of information to third parties may be part of an agreement that the agents make and possibly depends on the privacy policies of the involved agents. For example, an agent may agree to disclose part of its knowledge to another trusted agent, but disagree with the fact that the other agent may reveal its identity to a third party that it may not *a priori* know or trust. In this case, the latter two strategies are not acceptable, and the agents should resort to one of the first two strategies. Overall, integrating privacy policies, and choosing the appropriate strategy based on these policies is an interesting issue, which we plan to analyze in future extensions of this work.

8.2. Combining different strategies in a MCS

As we argued above, in a MCS each agent based on its computational capabilities and privacy policy may choose a different strategy for conflict resolution. In Sections 5 and 6 we discussed how distributed query evaluation is performed when all agents follow the same strategy. An interesting question is whether and how query evaluation can be performed in a MCS in which each agent may follow a different strategy.

Imagine, for example, a MCS in which agent A follows the third strategy (emphPropagating Mapping Sets), agent B follows the first strategy (*Single Answers*), while agent C has chosen the second strategy (*Strict-Weak Answers*). Suppose that A queries B about b_1, and that the truth value of b_1 depends on the truth value of c_1, which is derived strictly in C. In this scenario, the query evaluation algorithms that we described in Sections 5 and 6 cannot be directly applied for the following reasons:

– The answer that C returns for c_1 ($str(true)$) is meaningless for B.
– Assume that B manages to compute and return to A a positive ($true$) answer for b_1. As B follows the first strategy, it will return an empty set as the Supportive Set for b_1. A, which follows the third strategy, will assume that this answer is based on the strict local knowledge of B, which is not true.

As it becomes obvious from the example, enabling different agents to choose and follow different strategies requires that each agent is also aware of the strategies that the other agents that it interacts with follow, so that it can appropriately *translate* and evaluate the information that it imports from them. But even in this case, there are still some issues that must be handled. One of them is how to compare answers that are returned from agents following different strategies. In the example described above, suppose that A also queries another agent, D, about the truth value of d_1 and that D follows a different strategy from B, e.g. *Propagating Mapping Sets*. Suppose that D returns *true* as an answer for d_1 and a non-empty Supportive Set. In case the answers for b_1 and d_1 cause a conflict in the knowledge base of A, A must compare these answers according to its strategy (*Propagating Mapping Sets*). However, being aware that B follows the first strategy, A does not have information about whether the answer for b_1 is based on the local knowledge of B, or which other agents are involved in the derivation of this answer. As a consequence it cannot directly compare the answers for b_1 and d_1. Overall, to support MCS with agents following different strategies, we should consider how to deal with cases that information received from two or more agents following different strategies cause global conflicts in the system. It is a matter of further examination whether in such cases we should take into account not only the identities of the agents that knowledge is imported from but also the strategies followed by those agents.

9. Conclusion

This paper proposes a totally distributed approach for reasoning in Ambient Intelligence environments based on the Multi-Context Systems paradigm. To handle inconsistency in the distributed context knowledge, we added non-monotonic features in Multi-Context Systems. The proposed model uses local rule theories of Defeasible Logic to express local knowledge, defeasible rules for the definition of mappings, and a preference ordering on the system contexts to resolve conflicts caused by the interaction of distributed theories through the mappings. We also described four alternative strategies that use context and preference information for conflict resolution, and analyzed their formal properties with respect to termination, complexity and the possibility to create an equivalent global defeasible theory from the distributed contexts. We demonstrated the use of the algorithms in a use case scenario from the Ambient Intelligence domain. Finally, we described the implementation of the four strategies in a simulated peer-to-peer environment, which we used to evaluate the strategies with respect to their computational overhead. The obtained results highlight the tradeoff between the extent of context information that is exchanged between the ambient agents in order to evaluate the quality of imported knowledge and the computational complexity of the algorithms that implement the four strategies. Part of our ongoing work includes:

1. Studying the relation between our reasoning approach and loop checking variants of Defeasible Logic, such as those described in (Nute, 1997; Nute, 2001).
2. Extending our reasoning methods to support MCS with agents following different strategies.
3. Studying privacy issues that naturally arise in the Ambient Intelligence domain.
4. Implementing the algorithms in Logic Programming, using the equivalence with Defeasible Logic, and the well-studied translation of defeasible knowledge into logic programs under Well-Founded Semantics (Antoniou et al., 2006).
5. Implementing the described scenarios in real Ambient Intelligence environments with contexts lying on a variety of stationary and mobile devices (such as PDAs or cell phones).
6. Studying more applications in the Ambient Intelligence domain as well as in other domains with similar requirements such as Social Networks and the Semantic Web, where the theories may represent ontological context knowledge, policies and regulations.

References

Adjiman, P., Chatalic, P., Goasdoue', F., Rousset, M.-C. and Simon, L. (2006), 'Distributed Reasoning in a Peer-to-Peer Setting: Application to the Semantic Web', *Journal of Artificial Intelligence Research* **25**, 269–314.

Agostini, A., Bettini, C. and Riboni, D. (2005), Loosely Coupling Ontological Reasoning with an Efficient Middleware for Context-awareness, *in* 'Proceedings of MobiQuitous 2005', pp. 175–182.

Antoniou, G., Billington, D., Governatori, G. and Maher, M. (2006), 'Embedding defeasible logic into logic programming', *Theory Pract. Log. Program.* **6**(6), 703–735.

Antoniou, G., Billington, D., Governatori, G. and Maher, M. J. (2001), 'Representation results for defeasible logic', *ACM Trans. Comput. Logic* **2**(2), 255–287.

Bernstein, P. A., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L. and Zaihrayeu, I. (2002), Data Management for Peer-to-Peer Computing : A Vision, *in* 'WebDB', pp. 89–94.

Bikakis, A., Patkos, T., Antoniou, G. and Plexousakis, D. (2008), A Survey of Semantics-based Approaches for Context Reasoning in Ambient Intelligence, *in* M. Mühlhäuser, A. Ferscha and E. Aitenbichler, eds, 'Constructing Ambient Intelligence', Communications in Computer and Information Science, Springer, pp. 14–23.

Binas, A. and McIlraith, S. A. (2007), Exploiting Preferences over Information Sources to Efficiently Resolve Inconsistencies in Peer-to-peer Query Answering, *in* 'AAAI 2007 Workshop on Preference Handling for Artificial Intelligence'.

Brewka, G., Roelofsen, F. and Serafini, L. (2007), Contextual Default Reasoning, *in* 'IJCAI', pp. 268–273.

Buvac, S. and Mason, I. A. (1993), Propositional Logic of Context., *in* 'AAAI', pp. 412–419.

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M. and Rosati, R. (2005), Inconsistency Tolerance in P2P Data Integration: an Epistemic Logic Approach, *in* 'DBPL-05', Vol. 3774 of *LNCS*, SV, pp. 90–105.

Calvanese, D., De Giacomo, G., Lenzerini, M. and Rosati, R. (2004), Logical Foundations of Peer-To-Peer Data Integration, ACM, pp. 241–251.

Casali, A., Godo, L. and Sierra, C. (2008), A Logical Framework to Represent and Reason about Graded Preferences and Intentions, *in* 'KR', pp. 27–37.

Chatalic, P., Nguyen, G. H. and Rousset, M.-C. (2006), Reasoning with Inconsistencies in Propositional Peer-to-Peer Inference Systems, *in* 'ECAI', pp. 352–356.

Chen, H., Finin, T. and Joshi, A. (2003), 'Semantic Web in a Pervasive Context-Aware Architecture', *Artificial Intelligence in Mobile System 2003* pp. 33–40.

Cristani, M. and Burato, E. (2009), 'Approximate solutions of moral dilemmas in multiple agent system', *Knowledge and Information Systems* **18**(2), 157–181.

Dastani, M., Governatori, G., Rotolo, A., Song, I. and van der Torre, L. (2007), Contextual Deliberation of Cognitive Agents in Defeasible Logic, *in* 'AAMAS', p. 148.

Forstadius, J., Lassila, O. and Seppanen, T. (2005), RDF-based model for context-aware reasoning in rich service environment, *in* 'PerCom 2005 Workshops', pp. 15–19.

Franconi, E., Kuper, G. M., Lopatenko, A. and Serafini, L. (2003), A Robust Logical and Computational Characterisation of Peer-to-Peer Database Systems, *in* 'DBISP2P', pp. 64–76.

Gandon, F. L. and Sadeh, N. M. (2004), 'Semantic web technologies to reconcile privacy and context awareness', *Journal of Web Semantics* **1**, 241–260.

Ghidini, C. and Giunchiglia, F. (2001), 'Local Models Semantics, or contextual reasoning=locality+compatibility', *Artificial Intelligence* **127**(2), 221–259.

Giunchiglia, F. and Serafini, L. (1994), 'Multilanguage hierarchical logics, or: how we can do without modal logics', *Artificial Intelligence* **65**(1).

Gong, L. (2001), 'JXTA: A Network Programming Environment', *IEEE Internet Computing* **5**(3), 88–95.

Governatori, G., Maher, M. J., Antoniou, G. and Billington, D. (2004), 'Argumentation Semantics for Defeasible Logic', *J. Log. and Comput.* **14**(5), 675–702.

Gu, T., Pung, H. K. and Zhang, D. Q. (2004), A Middleware for Building Context-Aware Mobile Services, *in* 'Proceedings of the IEEE Vehicular Technology Conference (VTC 2004)', Milan, Italy.

Halevy, A. Y., Ives, Z. G., Suciu, D. and Tatarinov, I. (2003), Schema Mediation in Peer Data Management Systems, *in* 'ICDE', p. 505.

Hatala, M., Wakkary, R. and Kalantari, L. (2005), '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.

Henricksen, K. and Indulska, J. (2004), Modelling and Using Imperfect Context Information, *in* 'Proceedings of PERCOMW '04', IEEE Computer Society, Washington, DC, USA, pp. 33–37.

Khushraj, D., Lassila, O. and Finin, T. (2004), sTuples: Semantic Tuple Spaces, *in* 'First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous04)', pp. 267–277.

Kofod-Petersen, A. and Mikalsen, M. (2005), ‘Representing and Reasoning about Context in a Mobile Environment’, *Revue d’Intelligence Artificielle* **19**(3), 479–498.

Korpipaa, P., Mantyjarvi, J., Kela, J., Keranen, H. and Malm, E.-J. (2003), ‘Managing Context Information in Mobile Devices’, *IEEE Pervasive Computing* **02**(3), 42–51.

Krummenacher, R., Kopecký, J. and Strang, T. (2005), Sharing Context Information in Semantic Spaces., *in* ‘OTM Workshops’, pp. 229–232.

Maher, M. J. (2002), A Model-Theoretic Semantics for Defeasible Logic., *in* ‘Paraconsistent Computational Logic’, pp. 67–80.

McCarthy, J. (1987), ‘Generality in Artificial Intelligence’, *Communications of the ACM* **30**(12), 1030–1035.

McCarthy, J. and Buvač, S. (1998), Formalizing Context (Expanded Notes), *in* A. Aliseda, R. van Glabbeek and D. Westerståhl, eds, ‘Computing Natural Language’, CSLI Publications, Stanford, California, pp. 13–50.

Nguyen, N. T. and Katarzyniak, R. (2009), ‘Actions and social interactions in multi-agent systems’, *Knowledge and Information Systems* **18**(2), 133–136.

Nute, D. (1997), *Defeasible Deontic Logic*, Kluwer, chapter Apparent Obligation, pp. 288–315.

Nute, D. (2001), Defeasible Logic, *in* ‘Proceedings of the 14th International Conference on Applications of Prolog’, pp. 87–114.

Patkos, T., Bikakis, A., Antoniou, G., Plexousakis, D. and Papadopouli, M. (2007), A Semantics-based Framework for Context-Aware Services: Lessons Learned and Challenges, *in* ‘Proceedings of 4th International Conference on Ubiquitous Intelligence and Computing (UIC-2007)’, Vol. 4611 of *LNCS*, Springer, pp. 839–848.

Ranganathan, A. and Campbell, R. H. (2003), ‘An infrastructure for context-awareness based on first order logic’, *Personal Ubiquitous Comput.* **7**(6), 353–364.

Resconi, G. and Kovalerchuk, B. (2009), ‘Agents’ model of uncertainty’, *Knowledge and Information Systems* **18**(2), 213–229.

Roelofsen, F. and Serafini, L. (2005), Minimal and Absent Information in Contexts, *in* ‘IJCAI’, pp. 558–563.

Sabater, J., Sierra, C., Parsons, S. and Jennings, N. R. (2002), ‘Engineering Executable Agents using Multi-context Systems’, *Journal of Logic and Computation* **12**(3), 413–442.

Serafini, L. and Bouquet, P. (2004), ‘Comparing formal theories of context in AI’, *Artificial Intelligence* **155**(1-2), 41–67.

Sinner, A., Kleemann, T. and von Hessling, A. (2004), Semantic User Profiles and their Applications in a Mobile Environment, *in* ‘Artificial Intelligence in Mobile Systems 2004’.

Toninelli, A., Montanari, R., Kagal, L. and Lassila, O. (2006), A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments, *in* ‘Proc. of 5th International Semantic Web Conference’, pp. 5–9.

Turhan, A.-Y., Springer, T. and Berger, M. (2006), Pushing Doors for Modeling Contexts with OWL DL a Case Study, *in* ‘PERCOMW ’06: Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops’, IEEE Computer Society, Washington, DC, USA.

Wang, X. H., Dong, J. S., Chin, C. Y., Hettiarachchi, S. R. and Zhang, D. (2004), ‘Semantic Space: an infrastructure for smart spaces’, *IEEE Pervasive Computing* **3**(3), 32–39.

Wang, X. H., Zhang, D. Q., Gu, T. and Pung, H. K. (2004), Ontology Based Context Modeling and Reasoning using OWL, *in* ‘PERCOMW ’04: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops’, IEEE Computer Society, Washington, DC, USA, p. 18.