Distributed Defeasible Reasoning in Multi-Context Systems

Antonis Bikakis and Grigoris Antoniou
Institute of Computer Science, FO.R.T.H.
Vassilika Vouton, P.O. Box 1385, GR 71110, Heraklion, Greece
{bikakis,antoniou}@ics.forth.gr

Abstract
Multi-Context Systems (MCS) are logical formalizations of distributed context theories connected through a set of mapping rules, which enable information flow between different contexts. Reasoning in MCS introduces many challenges that arise from the heterogeneity of contexts with respect to the language and inference system that they use, and from the potential conflicts that may arise from the interaction of context theories through the mappings. This study proposes a P2P reasoning model for MCS, which represents contexts as peer theories in a P2P system, mapping rules as defeasible rules (rules that can be defeated in the existence of adequate contrary evidence), and uses a preference relation (which, e.g., expresses trust information) to resolve the potential conflicts. It also provides a reasoning algorithm for query evaluation, analyzes its formal properties, and discusses alternative methods for conflict resolution, which differ in the type of information that they use to resolve the conflicts.

Motivation and Background
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 as a logical theory - a set of axioms and inference rules - that models local context knowledge. 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 consequences of local theories using the mappings. The most critical issues of contextual reasoning are: (a) the heterogeneity of contexts with respect to the language and inference system that they use; and (b) the potential conflicts that may arise from the interaction of the different contexts through the mappings. Our study mainly focuses on the second issue, by modeling the different contexts as peers in a P2P system, and performing some type of defeasible reasoning on the distributed peer theories.

The notions of context and contextual reasoning were first introduced in AI by McCarthy in (McCarthy 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 Buvac 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). The second formalism was the basis of two recent studies that were the first to deploy non-monotonic reasoning approaches in MCS: (a) the non-monotonic rule-based MCS framework, which supports default negation in the mapping rules allowing to reason based on the absence of context information, proposed in (Roelofsen and Serafini 2005); and (b) the multi-context variant of Default Logic (Brewka, Roelofsen, and Serafini 2007). The latter models the bridge relations between different contexts as default rules, and has the additional advantage that is closer to implementation due to the well-studied relation between Default Logic and Logic Programming. However, the authors do not provide specific reasoning algorithms (e.g. for query evaluation), and their model does not include the notion of priority, which we use for conflict resolution.

Our study also relates to several recent studies that focus on formal models and methods for reasoning in peer data management systems. A key issue in formalizing data-oriented P2P 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 P2P systems. (Calvanes 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. in (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 (Calvanes 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. The proposed query evaluation algorithm assumes that all peers share a common alphabet of constants, and does not model trust or priorities between the peers. The proposi-
tional P2P inference system proposed in (Chatalic, Nguyen, and Rousset 2006) deals with conflicts caused by mutually inconsistent information sources, by detecting them and reasoning without them. The main problem is the same, once again: To perform reasoning, the conflicts are not actually resolved using some external trust or priority information; they are rather isolated.

The reasoning model that we propose represents contexts as peer theories in a P2P system. Specifically, it considers peers that have independent knowledge, and that interact with existing, neighboring peers to exchange information. The internal knowledge is expressed in terms of rules, and knowledge is imported from other peers through mapping rules. Even if it is assumed that the theory of each peer is locally consistent, the same assumption will not necessarily hold for the global knowledge base. The unification of the local context theories may result in inconsistencies that are caused by the mapping rules. For example, a context theory \( A \) may import context knowledge from two different contexts \( B \) and \( C \), through two competing mapping rules. In this case, even if the three different contexts are locally consistent, their unification through the mappings defined by \( A \) may contain inconsistencies. To deal with this type of inconsistencies (global conflicts), we follow a non-monotonic approach; mapping rules are expressed as defeasible rules (rules that may be defeated in the existence of adequate contrary evidence), and priorities between conflicting rules are determined by the level of trust that each peer has in the other system peers.

The P2P reasoning model captures the three fundamental dimensions of contextual reasoning, as these were formulated in (Benerecetti, Bouquet, and Ghidini 2000), namely partiality, approximation, and perspective:
- **Partiality.** Each peer may not have immediate access to all available information, so a peer theory can be thought as a partial representation of the world.
- **Approximation** Each peer theory differs at the level of detail at which a portion of the world is represented.
- **Perspective** Each peer theory encodes a different point of view on the world.

Furthermore, the P2P paradigm enables us to model:
- Information flow between different contexts as message exchange between the system peers.
- Context changes using the dynamics of a P2P system.
- Confidence on the different context theories as trust between the system peers.

The rest of the paper is structured as follows: First, we formalize the problem. Then, we describe the reasoning algorithm and study its formal properties. Finally, we discuss three alternative approaches for conflict resolution, and conclude with the plans of our future work.

**Our Reasoning Approach**

Our approach models a multi-context framework as a P2P system \( P \), which is a collection of peer context theories:

\[
P = \{ P_i \}, i = 1, 2, ..., n
\]

Each system peer has a proper distinct vocabulary \( V_{P_i} \), and a unique identifier \( i \). Each local theory is a set of rules that contain only local literals (literals from the local vocabulary). These rules are of the form:

\[
r^m_i : a^1_i, a^2_i, ..., a^{n-1}_i \rightarrow a^n_i
\]

where \( i \) denotes the peer identifier. Local rules express strict (sound) knowledge and are interpreted in the classical sense: whenever the literals in the body of a local rule \( (a^1_i, a^2_i, ..., a^{n-1}_i) \) derive as consequences of the local theory, then so does the conclusion of the rule \( a^n_i \). Strict rules with empty body are used to express factual knowledge.

Each peer also defines mappings that associate literals from its own vocabulary (local literals) with literals from the vocabulary of other peers (foreign literals). The acquaintances of peer \( P_i \), \( ACQ(P_i) \) are the set of peers that at least one of \( P_i \)'s mappings involves at least one of their local literals. Mappings are modeled as defeasible rules (rules that can be defeated in the existence of adequate contrary evidence) of the form:

\[
r^m_i : a^1_i, a^2_i, ..., a^{n-1}_i \Rightarrow a^n_i
\]

The above mapping rule is defined by \( P_i \) and associates some of its own local literals with some of the literals defined by \( P_j, P_k \) and other system peers. Literal \( a^n_i \) is a local literal of \( P_i \).

Finally, each peer \( P_i \) defines a trust level order \( T_i \), which includes a subset of the system peers, and expresses the trust that \( P_i \) has in the other system peers. This is of the form:

\[
T_i = [P_k, P_l, ..., P_n]
\]

A peer \( P_k \) is considered more trusted by \( P_i \) than peer \( P_l \) if \( P_k \) precedes \( P_l \) in this list. The peers that are not included in \( T_i \) are less trusted by \( P_i \) than those that are part of the list.

We assume that the context theories are locally consistent, but this is not necessarily true for the global theory, which derives from the unification of local theories and mappings. The inconsistencies result from interactions between local theories and are caused by mappings. To resolve them, we use the available trust information from the system peers.

**The P2P-DR Algorithm**

\( P2P_{DR} \) is a distributed algorithm for query evaluation in Multi-Context Systems following the model that we described in the previous section. The specific reasoning problem that it deals with is: **Given a MCS \( P \), and a query about literal \( x_i \) issued to peer \( P_j \), find the truth value of \( x_i \) considering \( P_j \)'s local theory, its mappings and the context theories of the other system peers.** The algorithm may return two different answers: (a) \( Ans_{x_i} = Yes \) means that it has computed a positive truth value for \( x_i \), while (b) \( Ans_{x_i} = No \) is meant for a negative truth value. The algorithm parameters are:

- \( x_i \): the queried literal
- \( P_0 \): the peer that issues the query
- \( P_j \): the local peer
- \( SS_{x_i} \): the Supportive Set of \( x_i \) (a set of literals that is initially empty)
To do that it calls a local reasoning algorithm (local or its negation \(x\) described later in this section), which returns a positive answer, in case \(x\) derives from the local rules, or a negative answer in any other case. Below, we denote as \(R_s(x_i)\) the set of rules that support \(x_i\) (as their conclusion); and as \(R_e(x_i)\), the set of rules that contradict \(x_i\) (those that support \(\neg x_i\)).

The algorithm proceeds in four main steps. In the first step, the algorithm determines if the queried literal, \(x_i\), or its negation \(\neg x_i\) are consequences of \(P_i\)'s local rules. To do that it calls a local reasoning algorithm (local \_alg, described later in this section), which returns a positive answer, in case \(x_i\) derives from the local rules, or a negative answer in any other case. Below, we denote as \(R_s(x_i)\) the set of rules that support \(x_i\) (as their conclusion); and as \(R_e(x_i)\), the set of rules that contradict \(x_i\) (those that support \(\neg x_i\)).

**P2P DR**\((x_i, P_0, P_t, SS_{x_i}, CS_{x_i}, Hist_{x_i}, Ans_{x_i}, T_i)\)

```latex
\begin{align*}
\text{if } \exists r^m_i \in R_e(x_i) \text{ then} & \quad \text{localHist}_{x_i} \leftarrow [x_i] \\
& \quad \text{call local\_alg}([x_i, localHist_{x_i}, localAns_{x_i})] \\
& \quad \text{if localAns}_{x_i} = Yes \text{ then} \quad \text{return Ans}_{x_i} = \text{localAns}_{x_i} \text{ and terminate} \\
\end{align*}
```

If Step 1 fails, the algorithm collects, in the second step, the local and mapping rules that support \(x_i\). To check which of these rules can be applied, it checks the truth value of the literals in their body by issuing similar queries (recursive calls of the algorithm) to \(P_t\) or to the appropriate neighboring peers \(P_t \in ACQ(P_i)\). To avoid cycles, before each new query, it checks if the same query has been issued before, during the same algorithm call (using Hist). For each applicable supportive rule \(r_i\), the algorithm builds its Supportive Set \(SS_{x_i}\). The Supportive Set of a rule derives from the unification of the set of the foreign literals (literals that are defined by peers that belong in \(ACQ(P_i)\)) that are contained in the body of \(r_i\), with the Supportive Sets of the local literals that belong in the body of the same rule. In the end, in case there is no applicable rule \(SR_{x_i} = \{\}\), the algorithm returns a negative answer for \(x_i\) and terminates. Otherwise, it computes the Supportive Set of \(r_i\), \(SS_{x_i}\), as the strongest of the Supportive Sets of the applicable rules that support \(x_i\), and proceeds to the next step. The strongest Supportive Set is computed using the Stronger function (described later in this section), which applies the preference relation defined by \(P_t, T_i\), on the given sets.

\[
CR_{x_i} = \{\}\n\]

```latex
\begin{align*}
\text{if } \forall r^m_i \in R_e(x_i) \text{ do} & \quad SS_{x_i} = \{\} \\
\text{for all } b_i \in body(r^m_i) \text{ do} & \quad \text{return Ans}_{x_i} = Yes \text{ and SS}_{x_i}, \text{ and terminate} \\
\end{align*}
```

In the third step, in the same way with the previous step, the algorithm collects the rules that contradict \(x_i\) and builds the conflicting set of \(x_i\) \((CS_{x_i})\). In case there is no applicable rule that contradicts \(x_i\), the algorithm terminates by returning a positive answer for \(x_i\). Otherwise, it proceeds with the last step. Below, we denote as \(CR_{x_i}\) the set of the applicable rules that contradict (support the negation of) \(x_i\).
In the last step, the algorithm compares $SS_x$ and $CS_x$, using again the $\text{Stronger}$ function. If $SS_x$ is stronger, the algorithm returns a positive answer for $x_i$. In any other case (including the case that there is not enough trust information available to give priority to one of the competing rules), it returns a negative answer.

if $\text{Stronger}(SS_x, CS_x, T_i) = SS_x$ then 
return $Ans_{x_i} = \text{Yes}$ and $SS_x$
else 
return $Ans_{x_i} = \text{No}$
end if

The local reasoning algorithm $\text{local\_alg}$ is called by $P2P_{DR}$ to determine whether a literal is a consequence of the local rules of the theory. The algorithm parameters are:

- $x_i$: the queried literal
- $\text{localHist}_{x_i}$: the list of pending queries in $P_i$
- $\text{localAns}_{x_i}$: the local answer for $x_i$ (initially No)

$\text{local\_alg}(x_i, \text{localHist}_{x_i}, \text{localAns}_{x_i})$

for all $r_i' \in R_k(x_i)$ do
if $\text{body}(r_i') = \{\}$ then
return $\text{localAns}_{x_i} = \text{Yes}$ and terminate
else
for all $b_i \in \text{body}(r_i')$ do
if $b_i \in \text{localHist}_{x_i}$ then
stop and check the next rule
else
$\text{localHist}_{b_i} \leftarrow \text{localHist}_{x_i} \cup b_i$
call $\text{local\_alg}(b_i, \text{localHist}_{b_i}, \text{localAns}_{b_i})$
end if
end for
if for every $b_i$: $\text{localAns}_{b_i} = \text{Yes}$ then
return $\text{localAns}_{x_i} = \text{Yes}$ and terminate
end if
end if
end for

The $\text{Stronger}(S, C, T)$ function is used by $P2P_{DR}$ to check which of $S$ and $C$ sets is stronger, based on $T$ (the preference relation defined by the peer that the algorithm is called by). According to $T$, a literal $a_k$ is considered to be stronger than $a_l$ if $P_k$ precedes $P_l$ in $T$. The strength of a set is determined by the the weakest literal in this set.

$\text{Stronger}(S, C, T)$

$a^w \leftarrow a_k \text{ s.t. for all } a_i \in S: P_k \text{ does not precede } P_l \text{ in } T$
$b^w \leftarrow a_l \text{ s.t. for all } b_j \in C: P_l \text{ does not precede } P_j \text{ in } T$
if $P_k$ precedes $P_l$ in $T$ then
$\text{Stronger} = S$
else if $P_l$ precedes $P_k$ in $T$ then
$\text{Stronger} = C$
else
$\text{Stronger} = \text{None}$
end if

Below we demonstrate how the algorithm works through an example. In the system depicted in Figure 1, there are six peers, each one with its local context theory, and a query about $x_1$ is issued to $P_1$.

Neither $x_1$ nor $\neg x_1$ derive from $P_1$’s local theory, so the algorithm proceeds to the second step. It successively calls rules $r_{11}'$ and $r_{12}'$, and issues a query about $a_2$ to $P_2$. In $P_2$, $a_2$ does not derive from the local theory, and the algorithm successively calls the two rules that support $a_2$, $r_{21}'$ and $r_{22}'$. $c_2$, which is the only premise of $r_{22}'$, is not supported by any rule, so $r_{22}'$ is not applicable. To check if rule $r_{22}'$ can be applied, the algorithm must check literal $b_2$. $b_2$ is supported by two mapping rules; $r_{23}'$ and $r_{24}'$. To determine if these rules can be applied, the algorithms queries $P_5$ about $b_5$ and $P_6$ about $b_6$. $P_5$ and $P_6$ return positive answers for $b_5$ and $b_6$, respectively, as these literals are consequences of their local theories. As there is no rule in $P_3$ that contradicts $b_2$ or $a_2$, $P_2$ returns a positive answer for $b_2$ to $P_1$, and $P_1$ constructs the Supportive Set of $a_1$, which contains literal $a_2 (SS_{a_2} = \{a_2\})$. The next step is to check the only rule that contradicts $a_1$, rule $r_{13}'$. Using a similar process, the algorithm ends up with a conflicting set that contains literals $a_3$ and $a_4$ ($CS_{a_1} = \{a_3, a_4\}$). To compare $SS_{a_1}$ and $CS_{a_1}$, the algorithm uses the trust level order defined by $P_1$, $T_1$. Assuming that $T_1 = \{P_4, P_2, P_6, P_3, P_5\}$, $a_2$ and $a_3$ are respectively the weakest elements of $SS_{a_1}$ and $CS_{a_1}$, and $a_3$ is weaker than $a_2$. Consequently, $P_1$ computes a positive answer for $a_1$, and, as there is no rule that contradicts $x_1$, it eventually returns a positive answer for $x_1$.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{A System of Six Context Theories}
\end{figure}

Properties of the Algorithm

In this section we describe some formal properties of $P2P_{DR}$ with respect to its termination (Proposition 1), complexity (Propositions 2-3), and the possibility to create an equivalent unified defeasible theory from the distributed context theories (Theorem 1). The proofs for Propositions 1-3 and Theorem 1 are available in (Bikakis 2008). Proposition 1 holds as cycles are detected within the algorithm.

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

Prop. 2 is a consequence of two states that we retain for each peer, which keep track of the incoming and outgoing queries of the peer.

Proposition 2 The total number of messages that are exchanged between the system peers for the computation of a
single query is $O(n^2)$ (in the worst case that all peers have defined mappings with all the other system peers), where $n$ stands for the total number of system peers.

**Proposition 3** The computational complexity of the algorithm on a single peer is in the worst case $O(n^2 \times n^2 \times n_r)$, where $n$ stands for the total number of system peers, $n_l$ stands for the number of literals a peer may define, and $n_r$ stands for the total number of (local and mapping) rules that a peer theory may contain.

**Equivalent Unified Defeasible Theory**

The goal of the procedure that we describe below is to build a global defeasible theory $T_v(P)$, which produces the same results with $P2P_{DR}$. The existence of such theory will enable us to resort to centralized reasoning in cases that we want to avoid decentralized control. The procedure consists of the following steps:

1. The local rules of each peer theory are added as strict rules in $T_v(P)$.
2. The mapping rules of each peer theory are added as defeasible rules in $T_v(P)$.
3. For each pair of conflicting rules in $T_v(P)$, we add a priority relation using the $\text{Priorities}_{dl}$ process that we describe below.

The role of $\text{Priorities}_{dl}$ is to augment $T_v(P)$ with the required rule priorities considering the trust level orders of the system peers. The process takes as input a literal of the theory, say $x_i$, the strict and defeasible rules of $T_v(P)$ that support or contradict $x_i$ ($R[x_i], R[\lnot x_i]$), and the trust ordering of $P_i, T_i$, and returns the Supportive Set of $x_i$ ($S_{x_i}$), and augments $T_v(P)$ with the required priority relations. The algorithm follows three main steps: In the first step, it builds the Supportive Sets for the rules that support or contradict $x_i$. These sets are built in a similar way that $P2P_{DR}$ computes the Supportive Sets of the respective rules in the distributed theories, with one only difference: If there is a literal in the body of a rule that contains $w$ in its Supportive Set, then the algorithm assigns $\{w\}$ as the Supportive Set of the rule, meaning that this rule is inapplicable.

In the second step, $\text{Priorities}_{dl}$ collects all the pairs of applicable conflicting rules and adds suitable priority relations by applying the Stronger function on their Supportive Sets (using the trust level order of the peer that defined $x_i, T_i$).

In the final step, the algorithm computes the Supportive Set of $x_i$ using the following rules: (a) If there is no applicable supportive rule, it returns $\{w\}$; (b) If there is an applicable contradicting rule that is stronger than all applicable supportive rules, it returns $\{w\}$; and (c) In any other case it returns the Supportive Set of the strongest applicable rule, using the Stronger function and $T_i$.

\[
\text{Priorities}_{dl}(x_i, R[x_i], R[\lnot x_i], T_i, S_{x_i})
\]

\[
S_{x_i} \leftarrow \{w\}
\]

\[
\text{for all } r_j \in R[x_i] \cup R[\lnot x_i] \text{ do}
\]

\[
S_{r_j} \leftarrow \{w\}
\]

\[
\text{for all } a_i \in \text{body}(r_i) \cap V_i \text{ do}
\]

\[
\text{call } \text{Priorities}_{dl}(a_i, R[a_i], R[\lnot a_i], T_i, S_{a_i})
\]

\[
S_{r_i} \leftarrow S_{r_i} \cup S_{a_i}
\]

\[
\text{end for}
\]

\[
\text{for all } a_j \in \text{body}(r_i) \text{ do}
\]

\[
\text{call } \text{Priorities}_{dl}(a_j, R[a_j], R[\lnot a_j], T_j, S_{a_j})
\]

\[
\text{if } w \in S_{a_j} \text{ then}
\]

\[
S_{r_i} \leftarrow \{w\}
\]

\[
\text{stop and check next } r_i
\]

\[
\text{else}
\]

\[
S_{r_i} \leftarrow S_{r_i} \cup a_j
\]

\[
\text{end if}
\]

\[
\text{end for}
\]

\[
\text{end for}
\]

\[
\text{end for}
\]

\[
\text{end for}
\]

\[
\text{end for}
\]

\[
\text{end for}
\]

\[
\text{end if}
\]

To prove Theorem 1, which follows, we use the following definition:

**Definition 1** A MCS $P$ is acyclic iff there is no rule $r \in P$ such that the conclusion of $r$ may be used to prove a literal in the body of $r$.

**Theorem 1** The global defeasible theory $T_v(P)$, augmented with the priority relations derived from the application of $\text{Priorities}_{dl}$ on all literals of the theory, produces, under the proof theory of (Antoniou et al. 2001), the same results as the application of $P2P_{DR}$ on the distributed context theories of an acyclic MCS $P$.

The latter property, which shows the equivalence with a defeasible theory, enables resorting to centralized reasoning by collecting the distributed context theories in a central entity and creating an equivalent defeasible theory. The complexity of $\text{Priorities}_{dl}$ used for the derivation of the equivalent global theory is comparable with the complexity of $P2P_{DR}$. Via Theorem 1, $P2P_{DR}$ 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).

**Alternative Approaches for Conflict Resolution**

In the algorithm that we described, each queried peer is required to return a single positive/negative answer for the literal it is queried about. When a conflict arises, a peer uses its trust information, to evaluate the quality of the answers it receives. Each answer is indirectly assigned with the trust value of the peer that returns it. This is one (and rather
the simplest) of the many different approaches that one can think of for conflict resolution. Below, we describe three different approaches, which differ in the type of information that a peer may use to evaluate the quality of the answers that it receives, and to resolve the potential conflicting answers that two or more different peers may return. To clarify the differences, we use the system of peers that we described in Figure 1.

**Strict-Weak Answers**

In this second approach, we attempt to associate the quality of the answer not only with the trust level of the queried peer, but also with the confidence of the queried peer on the answer it returns. Specifically, we define two levels of quality for each positive answer: (a) the strict answers, which derive from the local context theories; and (b) the weak answers, which derive from the combination of the local theory with the mappings of the queried peer. In this approach the strength of an element in a Supportive Set is determined primarily by the type of answer returned for this element (strict answers are considered stronger than weak ones), and secondly by the rank of the peer in the trust level order of the querying peer.

In the system of peers depicted in Figure 1, \( P_3 \) and \( P_4 \) will return a strict positive answer for \( a_3 \) and \( a_4 \) respectively, as the two literals derive directly from their local theories, while for \( a_2, P_2 \) will return a weak answer, as it cannot be locally proved. So, in this version, \( SS_{a_2} = \{a_2\} \) is weaker than \( CS_{a_1} = \{a_3, a_4\} \), despite the fact that \( P_2 \) precedes \( P_4 \) in \( T_1 \), and the algorithm will compute a negative answer for \( a_1 \), and eventually for \( x_1 \) as well.

**Propagating Mapping Sets**

Another approach is to evaluate the quality of an answer based on the ranks of the peers that are involved in the computation of this answer in the trust order defined by the queried peer. To support this feature, a peer does not return a single positive/negative answer for a literal it is queried about, but augments this answer with the Supportive Set of the queried literal. The peer, which receives the answer, uses this set to evaluate the quality of the answer, but also to build the Supportive Sets of its local literals.

In the system of peers in the example (Figure 1), \( P_3 \) and \( P_4 \) return a positive answer along with an empty Supportive Set for \( a_3 \) and \( a_4 \) respectively, as the two literals derive directly from their local theories. In the same way, \( P_5 \) and \( P_6 \) return positive answers along with empty Supportive Sets for literals \( b_5 \) and \( b_6 \) respectively to \( P_2 \). To build the Supportive Set of \( b_2 \), \( P_2 \) compares the Supportive Sets of the two rules that support \( b_2 \) (\( r_{23}^{a_2} \) and \( r_{24}^{a_4} \)). Considering that \( T_2 = \{P_1, P_5, P_6\} \), \( P_2 \) computes and returns to \( P_1 \), \( SS_{b_2} = \{b_5\} \). The algorithm will eventually compute \( SS_{a_1} = \{a_2, b_5\} \) and \( CS_{a_1} = \{a_3, a_4\} \). Considering that \( T_1 = \{P_3, P_2, P_6, P_3, P_5\} \), \( b_5 \) is the weakest element of \( SS_{a_1} \) and \( a_3 \) is the weakest element of \( CS_{a_1} \), and \( b_5 \) is weaker than \( a_3 \), so the algorithm will eventually return a negative answer for \( a_1 \) and \( x_1 \).

**Complex Mapping Sets**

The main feature of the previous approach is that along with the truth value of the queried literal, a peer also returns its Supportive Set. This set describes the most trusted course of reasoning that leads to the computed answer. However, trust is subjective. The most trusted between two or more different courses will be different considering the different trust level orderings of two different peers. This last approach has the distinct feature that the most trusted course is not determined by the queried peer but by the peer that issues the query. To support this feature, when a peer is queried about one of its local literals, it returns its truth value along with its Supportive Set, which in this case describes all the different ways that can be applied to support this literal. In this version, the Supportive Set of a literal is actually a set of the Supportive Sets of all the rules that can be applied to support this literal.

In the system of peers of the example (Figure 1), in order to build \( SS_{a_2} \), the algorithm will use the Supportive Sets of both rules (\( r_{23}^{a_2} \) and \( r_{24}^{a_4} \)) that can be applied to support \( b_2 \). So, in this version, \( P_5 \) returns \( SS_{a_1} = \{b_5\} \), and \( P_6 \) computes \( SS_{a_1} = \{a_2, b_5\}, \{a_3, b_5\} \) and \( CS_{a_1} = \{ \{a_3, a_4\} \}. From the two different courses that lead to a positive truth value for \( a_1 \), as these are described in \( SS_{a_1} \), \( P_1 \) computes that the one described in the second set (\( \{a_3, b_5\} \)) is stronger, as \( P_6 \) precedes \( P_5 \) in \( T_1 \). Comparing this set with \( CS_{a_1} = \{a_3, a_4\} \), the algorithm will determine that \( CS_{a_1} \) is weaker, as its weakest element, \( a_3 \) is weaker than \( b_5 \) according to \( T_1 \). Consequently, \( P_1 \) will compute and return positive answers for \( a_1 \) and eventually for \( x_1 \).

**Conclusion**

In this study, we proposed a model for Multi-Context Systems that represents contexts as local rule theories in a P2P system and mappings as defeasible rules, and uses context and preference information to resolve the potential conflicts that arise from the interaction of contexts through the mappings. We also described a distributed algorithm for query evaluation in MCS and analyzed its formal properties. Finally, we informally presented three alternative strategies for conflict resolution, which differ in the extent of context information that each local theory exploits to resolve potential conflicts. Part of our ongoing and future work includes:

- Implementing the algorithm 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).
- Studying in more detail the three alternative strategies for conflict resolution that we described in the previous section. Specifically, we plan to study each different version of the algorithm with regard to its properties (termination, complexity, equivalent defeasible theory).
- Adding non-monotonic features in the local context theories to support uncertainty and ambiguity in the local context knowledge.
• Extending the model to support overlapping vocabularies, which will enable different contexts to use elements of common vocabularies (e.g. URIs).

• Studying applications in the Ambient Intelligence and Semantic Web domains, where the theories may represent ontological context knowledge (e.g. in DLP), policies and regulations.

References