Alternative Strategies for Conflict Resolution in Multi-Context Systems

Antonis Bikakis[1], Grigoris Antoniou[2], and Panayiotis Hassapis[3]

[1]Institute of Computer Science, FO.R.T.H., Greece, bikakis@ics.forth.gr

[2]Institute of Computer Science, FO.R.T.H., Greece, antoniou@ics.forth.gr

[3]Department of Computer Science, Athens University of Ecomomics and Business, chasapis@aueb.gr

Abstract Multi-Context Systems are logical formalizations of distributed context theories connected through mapping rules, which enable information flow between different contexts. Reasoning in Multi-Context Systems introduces many challenges that arise from the heterogeneity of contexts with regard 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. The current paper proposes four alternative strategies for using context and preference information to resolve conflicts in a Multi-Context Framework, in which contexts are modeled as rule theories, mappings as defeasible rules, and global inconsistency is handled using methods of distributed defeasible reasoning.

1 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 of 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 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 [10], 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 [7], [11]), and the Multi-Context Systems introduced in [9], which later became associated with the Local Model Semantics proposed in [8]. MCS have been argued to be most adequate with respect to the three dimensions of contextual reasoning, as these were formulated in [5] (*partiality*, *approximation*, and *proximity*), and have been shown to be technically more general than PLC [13]. Multi-Context Systems have also been the basis of two recent studies that were the first to deploy non-monotonic reasoning methods in MCS: (a) the non-monotonic rule-based MCS framework of [12], which supports default negation in the mapping rules allowing to reason based on the absence of context information; and (b) the multi-context variant of Default Logic proposed in [6], which models bridge relations between different contexts as default rules, handling cases of inconsistency in the imported knowledge. However, none of these approaches includes the notion of priority or preference, which could be potentially used for conflict resolution.

This paper focuses on the problem of global conflicts in Multi-Context Systems; namely the inconsistencies that may arise when importing conflicting information from two or more different contexts. Even if all context theories are locally consistent, we cannot assume consistency in the global knowledge base. The unification of local theories may result in inconsistencies caused by the mappings. 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. In previous work [3], we proposed a reasoning model that represents contexts as local rule theories and mappings as defeasible rules, and a distributed algorithm for query evaluation, which exploits context and preference information from the system contexts to resolve such conflicts. In this paper, we describe three alternative strategies for conflict resolution, which differ in the extent of context knowledge that system contexts exchange in order to be able to resolve the potential conflicts.

The rest of the paper is organized as follows. Section 2 describes the proposed representation model. Section 3 describes the four alternative strategies for global conflict resolution, and how they are implemented in four different versions of a distributed algorithm for query evaluation. Section 4 presents the results of simulation-based experiments that we conducted on the four strategies using a prototypical implementation of the algorithms in a Java-based P2P system. Finally, the last section summarizes and discusses the plans of our future work.

2 Representation Model

Our approach models a Multi-Context System P as a collection of distributed local rule theories P_i:

$$P = \{P_i\}, i = 1,...,n$$

Each context (local theory) P_i has a proper distinct vocabulary V_i and a unique identifier i. It is defined as a tuple (V_i, R_i, T_i), where V_i is the vocabulary used by P_i, R_i is a set of rules, and T_i is a preference order on P.

R_i consists of two sets of rules: the set of local rules (R_i^l), and the set of mapping rules (R_i^m). Local rules contain only local literals (literals from the local vocabulary, V_i) in their heads and bodies. They are of the form:

$$r_i^l : a_i^1, a_i^2, ..., a_i^{n-1} \rightarrow a_i^n$$

Local rules express local context knowledge. Local rules with empty body are used to express factual knowledge.

Mapping rules associate literals from V_i (*local literals*) with literals from the vocabulary of other contexts (*foreign literals*). A mapping rule may contain both local and foreign literals in its body, and a local literal in its head. Mapping rules are modelled as defeasible rules; namely they cannot be applied to support their conclusion if there is adequate contrary evidence. They have the following form:

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

The above mapping rule is defined by P_i, and associates some of its own local literals (e.g. a_i^1) with some of the local literals of P_j (a_j^2), P_k (a_k^{n-1}) and other system contexts. The head of the rule (a_i^n) contains a local literal of the theory that has defined the rule (P_i).

Finally, each context P_i defines a (total/partial) preference order T_i on P to express its confidence on the knowledge it imports from other contexts. This is of the form:

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

A context P_k is preferred by P_i to context P_l if P_k precedes P_l in this order. A total preference order enables resolving all potential conflicts that may arise from the interaction of contexts through their mapping rules. Partial ordering enables resolving only conflicts caused by the interaction of certain contexts. However, it is closer to the needs of real-world distributed applications, where distributed entities cannot always be aware of the quality of information imported by any available information source. In case of partial ordering, the contexts that are not included in T_i are equally preferred by P_i and less preferred than the contexts that are contained in T_i.

3 Four Alternative Strategies for Conflict Resolution

In this section, we describe four versions of a distributed algorithm, *P2P_DR*, for query evaluation in MCS. Each version implements a different strategy for conflict resolution and handles the following problem: *Given a MCS P, and a query about literal x_i issued to context P_i, find the truth value of x_i considering P_i's local theory, its mappings and the rule theories of the other system contexts.*

A common characteristic of the four strategies is that they all use context knowledge and preference information from the system contexts to resolve potential conflicts that may arise when importing information from two or more different sources. Their main difference is in the type and extent of information that the system contexts exchange to evaluate the quality of imported context information. To demonstrate their differences, we describe how each strategy is applied to the scenario depicted in Fig. 1.

P_1
$r_{11}^l : a_1 \rightarrow x_1$
$r_{12}^m : a_2 \Rightarrow a_1$
$r_{13}^m : a_3, a_4 \Rightarrow \neg a_1$

P_2
$r_{21}^l : c_2 \rightarrow a_2$
$r_{22}^l : b_2 \rightarrow a_2$
$r_{23}^m : b_5 \Rightarrow b_2$
$r_{24}^m : b_6 \Rightarrow b_2$

P_3
$r_{31}^l :\rightarrow a_3$

P_4
$r_{41}^l :\rightarrow a_4$

P_5
$r_{51}^l :\rightarrow b_5$

P_6
$r_{61}^l :\rightarrow b_6$

Figure 1: A MCS of 6 context theories

In this system, there are six context theories and a query about literal x_1 is issued to context P_1. To compute the truth value of x_1, P_1 has to import knowledge from P_2, P_3 and P_4. In case the three system contexts return positive truth values for a_2, a_3 and a_4 respectively, there will be a conflict about the truth value of a_1 caused by the two conflicting mapping rules r_{12} and r_{13}.

3.1 Single Answers

Single Answers requires each context to return only the truth value of the queried literal. When a context receives conflicting answers from two different contexts, it resolves the conflict using its preference order. The version of the distributed algorithm that implements this strategy (*P2P_DR*$_{SA}$) is called by a context P_i when it receives a query about one of its local literals (say x_i) and proceeds as follows:

1. In the first step, it determines whether the queried literal, x_i or its negation $\neg x_i$ derive from P_i's local rules, returning a positive or a negative truth value respectively.
2. If Step 1 fails, the algorithm collects, in the second step, the local and mapping rules that support x_i (as their conclusion). For each such rule, it checks the truth value of the literals in its body, by issuing similar queries (recursive calls of the algorithm) to P_i or to the appropriate contexts. To avoid cycles, before each new query, it checks if the same query has been issued before, during the same algorithm call. If the algorithm receives positive answers for all literals in the body of a rule, it determines that this rule is applicable and builds its Supportive Set SS_{r_i}. This derives from the union of the set of the foreign literals contained in the body of r_i, with the Supportive Sets of the local literals in the body of r_i. In the end, in case there is no applicable supportive rule, the algorithm returns a negative answer for x_i and terminates. Otherwise, it computes the Supportive Set of x_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. To compute the strength of a set of literals, $P2P\_DR_{SA}$ uses the preference order T_i. A literal a_k is considered stronger than literal b_l if P_k precedes P_l in T_i. The strength of a set is determined by the weakest element in the set.
3. In the third step, the algorithm collects and checks the applicability of the rules that contradict x_i (rules with conclusion $\neg x_i$). If there is no such applicable rule, it terminates by returning a positive answer for x_i. Otherwise, it computes the Conflicting Set of x_i, CS_{x_i}, as the strongest of the Supportive Sets of the applicable rules that contradict x_i.
4. In its last step, $P2P\_DR_{SA}$ compares the strength of SS_{x_i} and CS_{x_i} using T_i to determine the truth value of x_i. If SS_{x_i} is stronger, the algorithm returns a positive truth value. Otherwise, it returns a negative one.

In the system of Fig. 1, $P2P\_DR_{SA}$ fails to produce a local answer for x_1. In the second step, it attempts to use P_1's mapping rules. The algorithm eventually receives positive answers for a_2, a_3 and a_4, and resolves the conflict for a_1 by comparing the strength of the Supportive Sets of the two conflicting rules, r_{12} and r_{13}. Assuming that $T_1=[P_4,P_2,P_6,P_3,P_5]$, it determines that $SS_{r_{12}}=\{a_2\}$ is stronger than $SS_{r_{13}}=\{a_3, a_4\}$ and returns positive answer for a_1 and eventually for x_1.

An analytical description of $P2P\_DR_{SA}$ is available in [3]. The same paper presents some formal properties of the algorithm regarding (a) its termination; (b) required number of messages ($O(n^2l)$, where n stands for the total number of contexts, whereas l stands for the number of literals a local vocabulary may contain); (c) computational complexity ($O(n^2l^2r)$, where r stands for the number of rules a context theory may contain); and (d) the existence of a unified defeasible theory that produces the same results with the distributed algorithm under the proof theory of Defeasible Logic [1].

3.2 Strength of Answers

The Strength of Answers strategy requires the queried context to return, along with the truth value of the queried literal, information about whether this value derives from its local theory or from the combination of the local theory and its mappings. To support this feature, the second version of the algorithm, $P2P\_DR_{SWA}$, supports two types of positive answers: (a) a *strict* answer indicates that a positive truth value derives from local rules only; (b) a *weak* answer indicates that a positive truth value derives from a combination of local and mapping rules. The querying context evaluates the answer based not only on the context that returns it but also on the type of the answer. This version follows the four main steps of $P2P\_DR_{SA}$ but with the following modifications:

a) A Supportive/Conflicting Set (of a rule or of a literal) is not a set of literals, but a set of the answers returned for these literals.
b) The strength of an element in a Supportive/Conflicting Set is determined primarily by the type of answer computed by the algorithm (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.

Given these differences, the execution of $P2P\_DR_{SWA}$ in the system depicted in Figure 1, produces the following results: The Supportive Sets of rules r_{12} and r_{13} are respectively: $SS_{r_{12}} = \{weak_{a_2}\}$, $SS_{r_{13}} = \{strict_{a_3}, strict_{a_4}\}$ (the truth values of a_3 and a_4 derive from the local theories of P_3 and P_4 respectively, while P_2 has to use its mappings to compute the truth value of a_2) and $SS_{r_{13}}$ is computed to be stronger than $SS_{r_{12}}$. Eventually, the algorithm computes negative truth values for a_1 and x_1.

3.3 Propagating Supportive Sets

The main feature of *Propagating Supportive Sets* is that along with the truth value of the queried literal, the queried context returns also its Supportive Set. The algorithm that implements this strategy, $P2P\_DR_{PS}$, differs from $P2P\_DR_{SA}$ only in the construction of a Supportive Set; in this case, the Supportive Set of a rule derives from the union of the Supportive Sets of all (local and foreign) literals in its body.

In the MCS depicted in Figure 1, $P2P\_DR_{PS}$ when called by P_2 to compute the truth value of a_2, and assuming that $T_2=[P_5,P_6]$, returns a positive value and its Supportive Set $SS_{a_2} = \{b_5\}$. The answers returned for literals a_3 and a_4 are both positive values and empty Supportive Sets (they are locally proved), and $P2P\_DR_{PS}$ called by P_1 computes $SS_{r_{12}} = \{a_2,b_5\}$ and $SS_{r_{13}} = \{a_3,a_4\}$. Using $T_1=[P_4,P_2,P_6,P_3,P_5]$, $P2P\_DR_{PS}$ determines that $SS_{r_{13}}$ is *stronger* than $SS_{r_{12}}$ (as both P_3 and P_4 precede P_5 in T_i), and eventually computes negative values for a_1 and x_1.

3.4 Complex Supportive Sets

Complex Supportive Sets, similarly with *Propagating Supportive Sets*, requires the queried context to return the Supportive Set of the queried literal along with its truth value. In the case of *Propagating Supportive Sets*, the Supportive Set is a set of literals that describe the most *preferred* (by the queried context) chain of reasoning that leads to the derived truth value. In the case of *Complex Supportive Sets*, the Supportive Set is actually a set of sets of literals; each different set describes a different chain of reasoning that leads to the computed truth value. The context that resolves the conflict determines the most *preferred* chain of reasoning using its own preference order. $P2P\_DR_{CS}$, differs from $P2P\_DR_{SA}$ as follows:

a) The Supportive Set of a rule derives from the product of the Supportive Sets of the literals in the body of the rule.
b) The Supportive Set of a literal derives from the union of the Supportive Sets of the applicable rules that support it.
c) Comparing the Supportive Set and the Conflicting Set of a literal requires comparing the *strongest* sets of literals contained in the two sets.

In the system of Figure 1, $P2P\_DR_{CS}$ called by P_2 computes a positive truth value for a_2 and $SS_{a_2} = \{\{b_5\},\{b_6\}\}$. When called by P_3 and P_4, $P2P\_DR_{CS}$ returns positive truth values and empty Supportive Sets for a_3 and a_4 respectively, while when called by P_1, it computes $SS_{r_{12}}=\{\{a_2,b_5\},\{a_2,b_6\}\}$ and $SS_{r_{13}}=\{\{a_3,a_4\}\}$. Using $T_1=[P_4,P_2,P_6,P_3,P_5]$, $P2P\_DR_{CS}$ determines that $\{a_2,b_6\}$ is the strongest set in $SS_{r_{12}}$, and is also stronger than $\{a_3,a_4\}$ (as P_6 precedes P_3 in T_1). Consequently, it returns a positive answer for a_1 and eventually a positive answer for x_1 as well.

Analytical descriptions of the three latter algorithms, as well as some results on their formal properties are omitted due to space limitations. Briefly, the three algorithms share the same properties regarding termination, number of messages, and the existence of an equivalent unified defeasible theory with $P2P\_DR_{SA}$. The computational complexity of second and third strategy is similar to $P2P\_DR_{SA}$; the fourth strategy imposes a much heavier computational overhead (exponential to the number of literals defined in the system).

4 Prototypical Implementation & Experimental Evaluation

In order to evaluate the four strategies, we implemented the respective versions of *P2P_DR* and a P2P system simulating the Multi-Context framework in Java.

4.1 Implementation

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 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 consists of 5 packages: *agencies, logic, knowledge, network, peerlib*. The *agencies* package contains the classes that implement the input file parsers and those that implement the four algorithms. The *logic* package contains the classes the represent (in memory) the literals and rules. The *knowledge* package includes the *KnowledgeBase* class; a Singleton class that stores the local and mapping rules, the preference orders and any other required information. 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 (e.g. the *Pipe* class).

4.2 Setup of the Experiments

The goal of the experiments was to compare the four strategies in terms of actual computational time spent by a peer to evaluate the answer to a single query. Using a tool that we built for the needs of the experiments, we created theories that correspond to the worst case that the computation of a single query requires computing the truth value of all literals from all system nodes. The test theories that we created have the following form:

$$
\begin{aligned}
& r^m_1 : a_2, a_3, \ldots, a_n \Rightarrow a_0 \\
& \quad \ldots \\
& r^m_{n/2} : a_1, \ldots a_{n/2-1}, a_{n/2+1} \ldots, a_n \Rightarrow a_0 \\
& r^m_{n/2+1} : a_1, \ldots a_{n/2}, a_{n/2+2} \ldots, a_n \Rightarrow \neg a_0 \\
& \quad \ldots \\
& r^m_n : a_1, a_2, \ldots, a_{n-1} \Rightarrow \neg a_0
\end{aligned}
$$

The above mapping rules are defined by P_0 and associate the truth value of a_0 with the truth value of the literals from n other system peers. Half of them support a_0 as their conclusion, while the remaining rules contradict a_0. In case the truth values returned for all foreign literals $a_1, a_2, \ldots, a_n$ are all positive then all mapping rules are applicable and are involved in the computation of the truth value of a_0.

To exclude the communication overhead from the total time spent by P_0 to evaluate the truth value of a_0, we filled a local cache of P_0 with appropriate answers for all the foreign literals. Specifically, for all strategies we used positive truth values for all foreign literals. For the second strategy (*Strength of Answers*),

the type of positive answer (*strict* or *weak*) was chosen randomly for each literal, and for the last two strategies we used Supportive Sets that involve all literals.

For each version of the algorithm we tested six experiments with a variant number of system peers (n): 10, 20, 40, 60, 80, and 100. The test machine was an Intel Celeron M at 1.4 GHz with 512 MB of RAM.

4.3 Results

Table 1 shows in msec the computation time for each version of *P2P_DR*. In the case of $P2P\_DR_{CS}$, we were able to measure the computation time only for the cases n = 10, 20, 40; in the other cases the test machine ran out of memory.

The computation time for the first three strategies is proportional to the square of the number of system peers. The fourth strategy requires much more memory space and computation time (almost exponential to the number of peers), which make it inapplicable in cases of very dense systems. The results also highlight the tradeoff between the computational complexity and the extent of context information that each strategy exploits to evaluate the confidence in the returned answers.

Table 1: Computation time for the four versions of *P2P_DR*

# peers (n)	$P2P\_DR_{SA}$	$P2P\_DR_{SWA}$	$P2P\_DR_{PS}$	$P2P\_DR_{CS}$
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	-

5 Conclusions and Future Work

In this paper, we proposed a totally distributed approach for reasoning with mutually inconsistent rule theories in Multi-Context Systems. The proposed model uses rule theories to express local context knowledge, defeasible rules for the definition of mappings, and a preference order to express confidence in the imported context information. We also described four strategies that use context and preference information for conflict resolution, and which differ in the extent of context information exchanged between the system contexts, and described how each strategy is implemented in a different version of a distributed algorithm for query evaluation in Multi-Context Systems. 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 requirements. The obtained results highlight the tradeoff between the extent of context information exchanged between the contexts to evaluate the quality of the imported context and the computational load of the algorithms that implement the four strategies.

Part of our ongoing work includes: (a) Implementing the algorithms in Logic Programming, using the equivalence with Defeasible Logic [3], and the well-studied translation of defeasible knowledge into logic programs under Well-Founded Semantics [2]; (b) Adding non-monotonic features in the local context theories to express uncertainty in the local knowledge; (c) Extending the model to support overlapping vocabularies, which will enable different contexts to use elements of common vocabularies (e.g. URIs); and (d) Implementing real-world applications of our approach in the Ambient Intelligence and Semantic Web domains. Some initial results regarding the application of our approach in Ambient Intelligence are already available in [4].

References

1. Antoniou G., Billington D., Governatori G., Maher M.J.: Representation results for defeasible logic. ACM Transactions on Computational Logic 2(2):255-287, 2001.
2. Antoniou G., Billington D., Governatori G., Maher M.J.: Embedding defeasible logic into logic programming. Theory and Practice of Logic Programming 6(6):703-735, 2006.
3. Bikakis A., Antoniou G.: Distributed Defeasible Reasoning in Multi-Context Systems. In NMR'08, pp. 200-206, (2008)
4. Bikakis A., Antoniou G.: Distributed Defeasible Contextual Reasoning in Ambient Computing. In AmI'08 European Conference on Ambient Intelligence, pp. 258-375, (2008)
5. Benerecetti M., Bouquet P., Ghidini C.: Contextual reasoning distilled. JETAI 12(3): 279-305, 2000.
6. Brewka G., Roelofsen F., Serafini L.: Contextual Default Reasoning. In: IJCAI, pp. 268-273 (2007)
7. Buvac, S, Mason I.A.: Propositional Logic of Context. In AAAI, pp. 412-419, (1993).
8. Ghidini C., Giunchiglia F.: Local Models Semantics, or contextual reasoning = locality + compatibility. Artificial Intelligence, 127(2):221-259, 2001.
9. Giunchiglia F., Serafini L.: Multilanguage hierarchical logics, or: how we can do without modal logics. Artificial Intelligence, 65(1), 1994.
10. McCarthy J.: Generality in Artificial Intelligence. Communications of the ACM, 30(12):1030-1035, 1987.
11. McCarthy J., Buvac S.: Formalizing Context (Expanded Notes). Aliseda A., van Glabbeek R., Westerstahl D. (eds.) Computing Natural Language, pp. 13-50. CSLI Publications, Stanford (1998)
12. Roelofsen F, Serafini L.: Minimal and Absent Information in Contexts. In IJCAI, pp. 558-563, (2005).
13. Serafini L., Bouquet P.: Comparing formal theories of context in AI. Artificial Intelligence, 155(1-2):41-67, 2004.