Distributed Reasoning with Conflicts in a Multi-Context Framework

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

Introduction
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 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 challenges of contextual reasoning are: (a) the heterogeneity of local context theories; and (b) the potential conflicts that may arise from the interaction of 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.

Two recent studies that deploy non-monotonic reasoning approaches in Multi-Context Systems are the nonmonotonic rule-based MCS framework, which supports default negation in the mapping rules, proposed in (Roelofsen and Serafini 2005), and the multi-context variant of Default Logic presented in (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, leaving some practical issues, such as the integration of priority information, unanswered.

Our study also relates to several studies that are focused on the semantic characterization of mappings in peer data management systems. Among them, (Franconi et al. 2003), (Calvanese et al. 2005), and (Chatalic, Nguyen, and Rousset 2006) are the most prominent that deal with conflicts caused by mutually inconsistent information sources, by detecting them and reasoning without them. A common deficiency of the two latter studies is that the conflicts are not actually resolved using some external trust or priority information, but they are rather isolated.

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 node has a proper distinct vocabulary \( V_i \) 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:
\[
r^i_1: a^1_i, a^2_i, ... a^{n-1}_i \rightarrow a^n_i
\]
where \( i \) denotes the node identifier.

Each node also defines mappings that associate literals from its own vocabulary (local literals) with literals from the vocabulary of other peers (remote literals). The acquaintances of node \( P_i \), \( ACQ(P_i) \), are the set of peers that at least one of \( P_i \)’s mappings involves at least one of their local literals. The mappings are rules of the form:
\[
r^{m}_i: a^1_i, a^2_j, ... a^{n-1}_k \Rightarrow a^n
\]

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 nodes. Finally, each node \( P_i \) defines a trust level order \( T_i \), which includes a subset of the system nodes, and expresses the trust that \( P_i \) has in the other system nodes.

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 nodes.

Problem Statement  Given a peer-to-peer system \( P \), and a query about literal \( x_i \) issued to peer \( P_i \), find the truth value of \( x_i \) considering \( P_i \)’s local theory, its mappings and the theories of the other system nodes.
The $P2P\_DR$ Algorithm The algorithm follows four main steps. The first one involves checking if the queried literal ($x_i$), or its negation ($\neg x_i$), are local consequences of $P_i$'s local theory. If not, the algorithm collects, in the second step, the local and mapping rules that support $x_i$. 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 neighboring nodes. To avoid cycles, before each new call, the algorithm checks if the same query has been issued before, during the same algorithm call. In the end of this step, the algorithm builds the mapping supportive set of $x_i$: this contains the set of foreign literals (literals that are defined by peers that belong in $AC(Q(P_i))$) that are contained in the body of those $P_i$'s mapping rules, which can be applied to prove $x_i$ in the absence of any contradictions. In the third step, in the same way with the second step, the algorithm collects the rules that contradict $x_i$ and builds the conflicting set of $x_i$. In the last step, the algorithm determines the truth value of $x_i$ by comparing the supportive and conflicting sets. To compare two mapping sets, a peer $P_i$ uses its trust level order, $T_i$. According to this, a literal $d_k$ is considered to be stronger than $b_l$ from $P_i$'s viewpoint if $P_k$ precedes $P_i$ in $T_i$. Below, we demonstrate how the algorithm works through the example depicted in Figure 1. A more detailed description of the algorithm is available at http://www.csd.uoc.gr/~bikakis/P2P2DR.pdf.

In the MCS system depicted in Figure 1, consider that 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 $r_{11}^l$, $r_{12}^l$ and $r_{13}^m$ and issues a query about $b_2$ to $P_2$. In $P_2$, two rules support $b_2$: $r_{21}^l$ and $r_{22}^l$. $c_2$, which is the only premise of $r_{21}^l$, is not supported by any rule, so $r_{21}^l$ is not applicable. To check if rule $r_{22}^l$ can be applied, the algorithm successively calls $r_{23}^m$ and issues a query about $d_5$ to $P_3$. $d_5$ is locally proven, so $P_3$ returns a positive answer for $d_5$. The algorithm, then, constructs the supportive set for $b_2$, which contains literal $d_5$ ($SS_{b_2} = \{d_5\}$). The next step is to check the only rule that contradicts $b_2$, rule $r_{23}^m$. Using a similar process, the algorithm ends up with a conflicting set that contains literals $b_3$ and $b_4$ ($CS_{b_2} = \{b_3, b_4\}$). To compare $SS_{b_2}$ and $CS_{b_2}$, the algorithm uses the trust level order defined by $P_2$. $T_2$. Assuming that $P_3$ precedes $P_5$ and $P_3$ precedes $P_5$ in $T_2$, $d_5$ and $b_4$ are respectively the weakest elements of $SS_{b_2}$ and $CS_{b_2}$, and $d_5$ is weaker than $d_5$. Consequently, $P_3$ returns a positive answer for $b_2$, and $P_1$ eventually returns a positive answer for $x_1$.

Some interesting properties of $P2P\_DR$ are:

- **Termination.** The algorithm is guaranteed to terminate returning either a positive or a negative answer for the queried literal (due to cycle detection).
- **Number of Messages.** With the addition of two states, which keep track of the incoming and outgoing queries of each system node, we can reduce the total number of messages that are exchanged between the system nodes for the computation of a single query to $O(n^2)$ (in the worst case that all nodes have defined mappings with all the other system nodes), where $n$ stands for total number of system nodes.
- **Single Node Complexity** The computational complexity of the algorithm on a single node is in the worst case $O(n^2 \times n_1^2 \times n_r)$, where $n_i$ stands for the number of literals a node 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.** Using a standard process, it is possible to unify the local context theories into a global defeasible theory, which produces the same results. In this theory, local rules are modeled as strict rules, mappings are modeled as defeasible rules, and trust information from the system nodes is used to derive priorities between conflicting rules.

**Planned Future Work**

Part of our ongoing work is to extend the algorithm to support defeasible local theories, overlapping vocabularies and non-Boolean queries. We also study alternative versions of the main algorithm, which differ in the way that a node evaluates the answers returned by its peers; for example we could associate the quality of an 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. Finally, we plan to study applications of our approach in the Ambient Intelligence and Semantic Web domains, where the theories may represent ontological context knowledge (Horn logic subset of OWL DL), policies and regulations.

**References**


