Distributed Reasoning with Conflicts in an Ambient Peer-to-Peer Setting

Antonis Bikakis and Grigoris Antoniou

Institute of Computer Science, FO.R.T.H., Vassilika Voutwn
P.O. Box 1385, GR 71110, Heraklion, Greece
{bikakis,antoniou}@ics.forth.gr

Abstract. In ambient environments, there coexist many different entities that collect, process, and change the available context information. Although they all share the same context, they face it from different viewpoints based on their perceptive capabilities, experiences and goals. Moreover, they are expected to use distinct vocabularies; they may even have different levels of sociality. This diversity raises additional research challenges in the study of Distributed Artificial Intelligence. In this paper, we present an algorithm for reasoning with distributed rule theories in an ambient setting. The algorithm models the participating agents as nodes in a peer-to-peer system, and considers the potential conflicts that may arise during the integration of the distributed theories taking into account some special characteristics of context knowledge and ambient agents.

1 Introduction

The study of ambient 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 the available context information and the special characteristics of the agents that provide and process this knowledge. Henricksen and Indulska in [1] 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 the data about a context property comes from multiple sources, the context information may become ambiguous. Imprecision is common in sensor-derived information, while erroneous context information arises as a result of human or hardware errors.

The agents that operate in an ambient environment are expected to have different goals, experiences and perceptive capabilities. They may use distinct vocabularies; they may even have different levels of sociality. Due to the highly dynamic and open nature of the environment (various entities join and leave the environment at random times), they are not able to know a priori all other entities that are present at a specific time instance nor can they communicate directly with all of them.

Considering these requirements, three main challenges of knowledge management in Ambient Intelligence are to enable:

1. Reasoning with the highly dynamic and ambiguous context data.
2. Managing the potentially huge piece of context data, in a real-time fashion, considering the restricted computational capabilities of some mobile devices.
3. Collective intelligence, by supporting information sharing, and distributed reasoning between the entities of the ambient environment.

So far, most pervasive computing frameworks have followed fully centralized approaches (e.g. [2–11]), while some others have employed models based on the *blackboard* and *shared memory* paradigms (e.g. [12–14]). Collecting the reasoning tasks in a central entity certainly has many advantages. It achieves better control, and better coordination between the participating entities. However, such solutions cannot meet the demanding requirements of ambient environments. The dynamics of the network and the unreliable and restricted (by the range of the transmitters) wireless communications inevitably lead to fully distributed solutions.

The goal of this study is to propose a distributed solution tailored to the special characteristics of ambient environments. The approach we propose to take models the agents of an ambient environment as nodes in a peer-to-peer system. Specifically, it considers nodes that have independent knowledge, and that interact with existing, *neighboring* nodes to exchange information. The internal knowledge is expressed in terms of rules, and knowledge is imported from other nodes through *bridging rules*.

Even if it is assumed that the theory of each node is locally consistent, the same assumption will not necessarily hold for the global knowledge base. The unification of the local theories, which model the viewpoints of the different nodes, may result in inconsistencies that are caused by the bridging rules. To deal with them, we follow a non-monotonic approach; bridging 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 node has on the other system nodes. In this way, the proposed approach manages to exploit the knowledge of every system node, and reason in a consistent and efficient manner, taking into account the viewpoint of each different node with regard to its context and cooperating peers.

The rest of the paper is structured as follows. Section 2 refers to the most prominent recent studies on reasoning in P2P data management systems and contextual reasoning. In Section 3, we present the algorithms that constitute our approach for reasoning with distributed rule theories. The conclusive section briefly describes the next steps of our work.

2 Related Work

Several recent studies have focused on developing formal models and methods for reasoning in peer-to-peer database systems. A key issue in formalizing data-oriented P2P systems is the semantic characterization of the *mappings* (bridging rules). One approach (followed in [15, 16]) is the first-order logic interpretation

of P2P systems. In [17], Calavanese *et al.* identifies several drawbacks with this approach, regarding modularity, generality and decidability, and proposes 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 [18] extends 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 by Calvanese *et al.* in [19], is based on nonmonontonic 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 propositional P2P inference system proposed by Chatalic *et al.* in [20] 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.

Relevant to our work are also some recent research studies that combine the fields of multi-context systems (MCS) and nonmonotonic reasoning. The first prominent work in this research line was conducted by Roelofsen and Serafini. They define in [21] a non-monotonic rule-based MCS framework, which contains default negation in the rules. The multi-context variant of Default Logic, introduced by Brewka *et al.* in [22] is a step further towards nonmonotonic contextual reasoning. Specifically, the authors propose to model the bridge relations between different contexts as *default rules*. The latter study 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 certain reasoning algorithms, leaving some practical issues, such as the integration of priority information, unanswered.

3 Our Approach

We propose modeling the agents of an ambient environment as nodes in a P2P system. This choice is not arbitrary. The P2P paradigm captures many critical properties of ambient settings:

1. Each different peer independently collects and processes in its own way the available context information.
2. Each peer may not have (immediate) access to all information sources.
3. The peers share their knowledge through messages with their neighboring nodes.
4. Each peer may not trust all the other peers at the same level.
5. Peers join and leave the system randomly.

Below, we define our P2P model, which captures local knowledge, mapping relations through which the nodes exchange information, and trust between the

system nodes. We also define the specific reasoning problem that we deal with, and describe the reasoning algorithms that we have developed.

3.1 Definitions

We assume a peer-to-peer system P as a collection of local theories:

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

Each 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:

$$r_i : a_i, b_i, ...k_i \rightarrow x_i$$

where i denotes the peer identifier.

Each peer 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 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. The mappings are rules of the form:

$$m_i : a_i, b_j, ...z_k \rightarrow x$$

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. Literal x may belong to whichever vocabulary of these system nodes. Finally, each peer defines a trust order T_i, which includes a subset of the system nodes.

3.2 Problem Statement

Given a peer-to-peer system P, and a query about literal x_i issued at 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.

We assume that the local theories are consistent, but this is not necessarily true for the case of the unified theory $T(P)$, which is the collection of the theories (local rules and mappings) of the system nodes. The inconsistencies result from interactions between local theories and are caused by mappings.

An example of such conflicts derives in the following system of theories:

P_1	P_2	P_3
$r_{11} : a_1 \rightarrow x_1$	$r_{21} : \rightarrow a_2$	$r_{31} : \rightarrow a_3$
$m_{11} : a_2 \rightarrow a_1$		
$m_{12} : a_3 \rightarrow \neg a_1$		

P_i's theory is locally consistent, but with the addition of the the two mapping rules (m_{11}, m_{12}), which associate the literals of P_1 with those of P_2 and P_3, a conflict about literal a_1 derives from the interaction of the three theories.

3.3 *P2P_DR* Algorithm

The algorithm follows four main steps. In the first step (lines 1-16), it uses P_i's local theory to prove x_i. If x_i or its negation, $\neg x_i$, derives from the peer's local theory, the algorithm terminates returning Yes/No respectively, without considering the peer's mappings or the theories of other peers in the system.

In the second step (lines 17-41), if neither x_i nor $\neg x_i$ derives from the local theory, the algorithm also uses P_i's mappings. It collects all the rules that support x_i. For each such rule, it checks the provability of the literals in its body. For each local/remote literal, it issues similar queries (recursive calls of the algorithm) to P_i (local literals) or to the appropriate P_i's acquaintances (remote literals). To avoid circles, before each new call, the algorithm checks if the same query has been issued before, during the same query evaluation process. At the end of this step, the algorithm builds the mapping supportive set of x_i; this contains the set of mapping (locally or remotely defined) rules that can be used to prove x_i in the absence of contradictions.

The third step (lines 42-66) involves the rules that contradict x_i. The algorithm builds the mapping conflicting set of x_i, by collecting the rules that support $\neg x_i$.

In the last step (lines 64-71), the algorithm decides about x_i by comparing the supportive and conflicting sets. To compare two mapping sets, a peer uses its trust order T_i. According to this order, one mapping rule m_k is considered to be stronger than m_l from P_i's viewpoint if P_i trusts P_k more than P_l. The strength of a mapping set is determined by the weakest rule in this set. In the followings, we denote as:

r_i^l: a local rule of P_i
r_i^m: a mapping rule of P_i
r_i^{lm}: a rule (local/mapping) of P_i
R^m: the set of all mapping rules
$R_s(x_i)$: the set of supportive rules for x_i
$R_c(x_i)$: the set of conflicting rules for x_i

When a node P_i receives a query about x_i, it runs the **P2P_DR** algorithm. The algorithm parameters are:
x_i: the queried literal
P_0: the peer that issued the query
P_i: the local node
SS_{x_i}: the set of supportive mappings for x_i (initially empty)
CS_{x_i}: the set of conflicting mappings for x_i (initially empty)
$Hist_{x_i}$: the list of pending queries of the form: $[x_1, ..., x_i]$
Ans_{x_i}: the answer returned for x_i (initially empty)

P2P_DR$(x_i, P_0, P_i, SS_{x_i}, CS_{x_i}, Hist_{x_i}, Ans_{x_i})$
1: **if** $\exists r_i^l \in R_s(x_i)$ **then**
2: $localHist_{x_i} \leftarrow [x_i]$

run $local\_alg(x_i, localHist_{x_i}, localAns_{x_i})$
if $localAns_{x_i} = Yes$ **then**
$Ans_{x_i} \leftarrow localAns_{x_i}$
terminate
end if
end if
if $\exists r_i^l \in R_c(x_i)$ **then**
$localHist_{x_i} \leftarrow [x_i]$
run $local\_alg(\neg x_i, localHist_{x_i}, localAns_{\neg x_i})$
if $localAns_{\neg x_i} = Yes$ **then**
$Ans_{x_i} \leftarrow \neg localAns_{\neg x_i}$
terminate
end if
end if
for all $r_i^{lm} \in R_s(x_i)$ **do**
$SS_{r_i} \leftarrow \{\}$
for all $b_t \in body(r_i^{lm})$ **do**
if $b_t \in Hist_{x_i}$ **then**
stop and check the next rule
else
$Hist_{b_t} \leftarrow Hist_{x_i} \bigcup b_t$
run $P2P\_DR(b_t, P_i, P_t, SS_{b_t}, CS_{b_t}, Hist_{b_t}, Ans_{b_t})$
if $Ans_{b_t} = No$ **then**
stop and check the next rule
else
$SS_{r_i} \leftarrow SS_{r_i} \bigcup SS_{b_t}$
end if
end if
end for
if $r_i^{lm} \in R^m$ **then**
$SS_{r_i} \leftarrow SS_{r_i} \bigcup r_i^{lm}$
end if
if $Stronger(SS_{r_i}, SS_{x_i}, T_i) = Yes$ **then**
$SS_{x_i} \leftarrow SS_{r_i}$
end if
end for
if $SS_{x_i} = \{\}$ **then**
return $Ans_{x_i} = No$ and terminate
end if
for all $r_i^{lm} \in R_c(x_i)$ **do**
$SS_{r_i} \leftarrow \{\}$
for all $b_t \in body(r_i^{lm})$ **do**
if $b_t \in Hist_{x_i}$ **then**
stop and check the next rule
else

```
48:          Hist_{b_t} ← Hist_{x_i} ⋃ b_t
49:          run P2P_DR(b_t, P_i, P_t, SS_{b_t}, CS_{b_t}, Hist_{b_t}, Ans_{b_t})
50:          if Ans_{b_t} = No then
51:             stop and check the next rule
52:          else
53:             SS_{r_i} ← SS_{r_i} ⋃ SS_{b_t}
54:          end if
55:       end if
56:    end for
57:    if r_i^{lm} ∈ R^m then
58:       SS_{r_i} ← SS_{r_i} ⋃ r_i^{lm}
59:    end if
60:    if Stronger(SS_{r_i}, CS_{x_i}, T_i) = Yes then
61:       CS_{x_i} ← SS_{r_i}
62:    end if
63: end for
64: if CS_{x_i} = {} then
65:    return Ans_{x_i} = Yes and SS_{x_i} and terminate
66: end if
67: if Stronger(SS_{x_i}, CS_{x_i}, T_i) = Yes then
68:    return Ans_{x_i} = Yes and SS_{x_i} and terminate
69: else
70:    Ans_{x_i} = No and terminate
71: end if
```

The $local\_alg(x_i, localHist_{x_i}, localAns_{x_i})$ is used to determine if x_i is a consequence of P_i's local theory. The algorithm parameters are:

x_i: the queried literal

$localHist_{x_i}$: the list of pending queries in P_i of the form: $[x_i^1, ..., x_i^m]$

$localAns_{x_i}$: the local answer returned for x_i (initially No)

local_alg$(x_i, localHist_{x_i}, localAns_{x_i})$

```
1: for all r_i^l ∈ R_s(x_i) do
2:    if body(r_i^l) = {} then
3:       return localAns_{x_i} = Yes
4:       terminate
5:    else
6:       for all b_i ∈ body(r_i^l) do
7:          if b_i ∈ localHist_{x_i} then
8:             stop and check the next rule
9:          else
10:            localHist_{b_i} ← localHist_{x_i} ⋃ b_i
11:            run local_alg(b_i, localHist_{b_i}, localAns_{b_i})
12:         end if
13:      end for
```

14: **if** for every b_i: $localAns_{b_i} = Yes$ **then**
15: $localAns_{x_i} \leftarrow Yes$
16: terminate
17: **end if**
18: **end if**
19: **end for**

The $Stronger(S, C, T_i)$ function is used by P_i to check if the S set of mappings is stronger than the C set of mappings based on P_i's trust level order, T_i.

Stronger(S, C, T_i)
1: $r_s^w \leftarrow r_s \in S$ s.t. $forall\ r_i \in S : r_s$ is not weaker than r_i (according to T_i)
2: $r_c^w \leftarrow r_c \in C$ s.t. $forall\ r_j \in C : r_c$ is not weaker than r_j (according to T_i)
3: **if** r_s^w is stronger than r_c^w **then**
4: $Stronger = Yes$
5: **else**
6: $Stronger = No$
7: **end if**

3.4 Algorithm Properties

The application of the proposed algorithms in real scenarios largely depends on some properties regarding its termination and complexity.

Termination. We assume that there are a finite number of nodes in the system, each of which with a finite number of literals in its vocabulary. As a consequence, there are a finite number of rules that a peer may define. If the algorithm did not terminate, it would have to make indefinite recursive calls, adding each time a new query to the history, without ever returning an answer or detecting a cycle. However, this is impossible, because: (a) the number of recursive calls is bounded by the total finite number of literals in the system; and (b) there can be a finite number of independent (with different history) algorithm calls. These are bounded by the total finite number of rules in the system. Consequently, the algorithm will eventually terminate.

Number of Messages. To reduce the complexity of the algorithm with regard to the number of messages that the system nodes have to exchange, and the computational overhead of the algorithm on each system node, we can make the following optimization: Each node is required to retain two states: (a) the state of the queries it has been requested to process, INC_Q; this contains tuples of the form (q_i, Ans_{q_i}), where q_i is the queried literal, and Ans_{q_i} is $true/false$ in the case the node has completed the computation, or *undetermined* otherwise; and (b) the state of the queries it has requested other peers to process, OUT_Q (of the same form). Before sending a query to one of its neighbors, a node checks

if the same query is in OUT_Q. If this is the case, it retrieves the answer stored in OUT_Q if this has the value $true/false$, or waits until the pending query returns a $true/false$ answer. When a new query is issued at a node, the node checks if the same query is in its INC_Q. If it is, the node returns the stored $true/false$ answer for that query if this has already been computed; otherwise, it suspends the new query until the pending query returns a $true/false$ answer. The space overhead of both states is proportional to the number of mappings that a node defines. The two states need to be updated every time a new query is issued at the system from an external source (we assume that the state of the network remains unchanged during the computation of each such query).

With these optimizations, each node will have to make at most one query for each of the remote literals that appear in the body of its mapping rules. In the worst case, that each peer has defined mappings that involve literals from all the other nodes in the system, and needs to apply all these mappings during a query evaluation, each peer will have to make $n \times n_l$ queries, where n is the number of system nodes and n_l is the maximum number of literals that a node may define. So, the total number of messages that need to be exchanged for the computation of a single query is in the worst case $n \times n \times n_l = O(n^2)$ (assuming that the number of nodes is the most critical parameter in the system).

4 Conclusion

We presented an approach for distributed reasoning in P2P settings, taking into account some special properties and constraints of context knowledge and ambient environments. The proposed reasoning algorithm models and reasons with potential conflicts that may arise during the integration of the distributed theories; to resolve these conflicts it uses trust information from the system nodes. We have already proved some desirable algorithm properties regarding its termination and complexity, and we are in the course of studying other properties, such as the computational complexity of the distributed algorithm on a single node. Other planned research directions of the same work are: (a) Study if there is an equivalent defeasible theory that derives from the unification of the distributed theories and produces the same results; (b) Extend the algorithm to support overlapping vocabularies; (c) Extend the algorithm to support defeasible local rules, and non-Boolean queries; and (d) Study applications in the Ambient Intelligence domain, where the theories may represent ontological knowledge (Horn logic subset of OWL DL), policies or regulations.

References

1. Henricksen, K., Indulska, J.: Modelling and Using Imperfect Context Information. In: Proceedings of PERCOMW '04, Washington, DC, USA, IEEE Computer Society (2004) 33–37
2. Chen, H., Finin, T., Joshi, A.: Semantic Web in a Pervasive Context-Aware Architecture. Artificial Intelligence in Mobile System 2003 (2003) 33–40

3. Forstadius, J., Lassila, O., Seppanen, T.: RDF-based model for context-aware reasoning in rich service environment. In: PerCom 2005 Workshops. (2005) 15–19
4. Patkos, T., Bikakis, A., Antoniou, G., Plexousakis, D., Papadopouli, M.: 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). (2007) accepted for publication
5. Gu, T., Pung, H.K., Zhang, D.Q.: A Middleware for Building Context-Aware Mobile Services. In: Proceedings of the IEEE Vehicular Technology Conference (VTC 2004), Milan, Italy (2004)
6. Wang, X.H., Dong, J.S., Chin, C.Y., Hettiarachchi, S.R., Zhang, D.: Semantic Space: an infrastructure for smart spaces. IEEE Pervasive Computing **3**(3) (2004) 32–39
7. Ranganathan, A., Campbell, R.H.: An infrastructure for context-awareness based on first order logic. Personal Ubiquitous Comput. **7**(6) (2003) 353–364
8. Gandon, F.L., Sadeh, N.M.: Semantic web technologies to reconcile privacy and context awareness. Journal of Web Semantics **1** (2004) 241–260
9. Toninelli, A., Montanari, R., Kagal, L., Lassila, O.: A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments. In: Proc. of 5th International Semantic Web Conference. (2006) 5–9
10. Kofod-Petersen, A., Mikalsen, M.: Representing and Reasoning about Context in a Mobile Environment. Revue d'Intelligence Artificielle **19**(3) (2005) 479–498
11. Hatala, M., Wakkary, R., Kalantari, L.: 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) (2005) 5–22
12. Khushraj, D., Lassila, O., Finin, T.: sTuples: Semantic Tuple Spaces. In: First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous04). (2004) 267–277
13. Krummenacher, R., Kopecký, J., Strang, T.: Sharing Context Information in Semantic Spaces. In: OTM Workshops. (2005) 229–232
14. Korpipaa, P., Mantyjarvi, J., Kela, J., Keranen, H., Malm, E.J.: Managing Context Information in Mobile Devices. IEEE Pervasive Computing **02**(3) (2003) 42–51
15. Bernstein, P.A., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., Zaihrayeu, I.: Data Management for Peer-to-Peer Computing : A Vision. In: WebDB. (2002) 89–94
16. Halevy, A.Y., Ives, Z.G., Suciu, D., Tatarinov, I.: Schema Mediation in Peer Data Management Systems. In: ICDE. (2003) 505
17. Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Logical Foundations of Peer-To-Peer Data Integration, ACM (2004) 241–251
18. Franconi, E., Kuper, G.M., Lopatenko, A., Serafini, L.: A Robust Logical and Computational Characterisation of Peer-to-Peer Database Systems. In: DBISP2P. (2003) 64–76
19. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Inconsistency Tolerance in P2P Data Integration: an Epistemic Logic Approach. In: DBPL-05. Volume 3774 of LNCS., SV (2005) 90–105
20. Chatalic, P., Nguyen, G.H., Rousset, M.C.: Reasoning with Inconsistencies in Propositional Peer-to-Peer Inference Systems. In: ECAI. (2006) 352–356
21. Roelofsen, F., Serafini, L.: Minimal and Absent Information in Contexts. In: IJCAI. (2005) 558–563
22. Brewka, G., Roelofsen, F., Serafini, L.: Contextual Default Reasoning. In: IJCAI. (2007) 268–273