Contextual Argumentation in Ambient Intelligence

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. The imperfect nature of context in Ambient Intelligence environments and the special characteristics of the entities that possess and share the available context information render contextual reasoning a very challenging task. Most current Ambient Intelligence systems have not successfully addressed these challenges, as they rely on simplifying assumptions, such as perfect knowledge of context, centralized context, and unbounded computational and communicating capabilities. This paper presents a knowledge representation model based on the Multi-Context Systems paradigm, which represents ambient agents as autonomous logic-based entities that exchange context information through mappings, and uses preference information to express their confidence in the imported knowledge. On top of this model, we have developed an argumentation framework that exploits context and preference information to resolve conflicts caused by the interaction of ambient agents through mappings, and a distributed algorithm for query evaluation.

1 Introduction

The study of Ambient Intelligence environments has introduced new research challenges in the field of Distributed Artificial Intelligence. These are mainly caused by the imperfect nature of context and the special characteristics of the entities that possess and share the available context information. [1] characterizes four types of imperfect context: *unknown*, *ambiguous*, *imprecise*, and *erroneous*. The agents that operate in such environments are expected to have different goals, experiences and perceptive capabilities, limited computation capabilities, and use distinct vocabularies to describe their context. Due to the highly dynamic and open nature of the environment and the unreliable wireless communications that are restricted by the range of transmitters, ambient agents do not typically know a priori all other entities that are present at a specific time instance nor can they communicate directly with all of them.

So far, Ambient Intelligence systems have not managed to efficiently handle these challenges. As it has been already surveyed in [2], most of them follow classical reasoning approaches that assume perfect knowledge of context, failing to deal with cases of missing, inaccurate or inconsistent context information. Regarding the distribution of reasoning tasks, a common approach followed in most systems assumes the existence of a central entity, which is responsible for collecting and reasoning with all the available context information. However, Ambient Intelligence environments have much more demanding requirements. The dynamics of the network and the unreliable and restricted wireless communications inevitably lead to the decentralization of reasoning tasks.

E. Erdem, F. Lin, and T. Schaub (Eds.): LPNMR 2009, LNCS 5753, pp. 30–43, 2009.

In this paper, we propose a totally distributed approach for contextual reasoning in Ambient Intelligence. We model an ambient environment as a *Multi-Context System*, and ambient agents as autonomous logic-based entities that exchange information through *mapping* rules and use preference information to evaluate imported knowledge. Then we provide a semantic characterization of our approach using *arguments*. The use of arguments is a natural choice in multi-agent systems and aims at a more formal and abstract description of our approach. Conflicts that arise from the interaction of mutually inconsistent sources are captured through *attacking* arguments, and conflict resolution is achieved by *ranking* arguments according to a preference ordering. Finally, we provide an operational model in the form of a distributed algorithm for query evaluation. The algorithm has been implemented in Java and evaluated in a simulated P2P system, and the results are available in [3]. Here, we focus more on its formal properties.

The rest of the paper is structured as follows. Section 2 presents background information and related work on contextual reasoning and preference-based argumentation systems. Section 3 presents the representation model, while section 4 describes its argumentation semantics. Section 5 presents the distributed algorithm and studies its properties. The last section summarizes the main results and discusses future work.

2 Background and Related Work

2.1 Multi-Context Systems

Since the seminal work of McCarthy [4] on *context* and *contextual reasoning*, two main formalizations have been proposed to formalize context: the *Propositional Logic of Context* (*PLC* [5]), and the *Multi-Context Systems* (*MCS*) introduced in [6], which later became associated with the *Local Model Semantics* [7]. MCS have been argued to be most adequate with respect to the three dimensions of contextual reasoning (*partiality*, *approximation*, *proximity*) and shown to be technically more general than PLC [8]. A MCS consists of a set of *contexts* and a set of inference rules (known as *mapping* rules) that enable information flow between different contexts. A context can be thought of as a logical theory - a set of axioms and inference rules - that models local knowledge. Different contexts are expected to use different languages, and although each context may be locally consistent, global consistency cannot be required or guaranteed.

The MCS paradigm has been the basis of two recent studies that were the first to deploy non-monotonic features in contextual reasoning: (a) the non-monotonic rule-based MCS framework [9], 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 (*ConDL* [10]), which additionally handles the problem of mutually inconsistent information provided by two or more different sources using default mapping rules. However, *ConDL* does not provide ways to model the quality of imported context information, nor preference between different information sources, leaving the conflicts that arise in such cases unresolved. The use of Multi-Context Systems as a means of specifying and implementing agent architectures has been recently proposed in [11], which proposes *breaking* the logical description of an agent into a set of contexts, each of which represents a different component of the architecture, and the interactions between these components are specified by means of bridge rules between the contexts.

Here, we follow a different approach; a context does not actually represent a logical component of an agent, but rather the viewpoint of each different agent in the system.

Peer data managements systems can be viewed as special cases of MCS, as they consist of autonomous logic-based entities (*peers*) that exchange local information using *bridge rules*. Two prominent recent works that handle the problem of peers providing mutually inconsistent information are: (a) the approach of [12], which is based on nonmonotonic epistemic logic; and (b) the propositional P2P Inference System of [13]. A major limitation of both approaches is that conflicts are not actually resolved using some external preference information; they are rather isolated. Our approach enables resolving such conflicts using a preference ordering on the information sources. Building on the work of [13], [14] proposed an argumentation framework and algorithms for inconsistency resolution in P2P systems using a preference relation on system peers. However, their assumptions of a single global language and a global preference relation are in contrast with the dimension of perspective in MCS. In our approach, each agent uses its own vocabulary to describe its context and defines its own preference ordering.

2.2 Preference-Based Argumentation Systems

Argumentation systems constitute a way to formalize non-monotonic reasoning, viz. as the construction and comparison of arguments for and against certain conclusions. A central notion in such systems is that of *acceptability* of arguments. In general, to determine whether an argument is acceptable, it must be compared to its counter-arguments; namely, those arguments that support opposite or conflicting conclusions. In preference-based argumentation systems, this comparison is enabled by a preference relation, which is either implicitly derived from elements of the underlying theory, or is explicitly defined on the set of arguments. Such systems can be classified into four categories. In the first category, which includes the works of [15] and [16], the preference relation takes into account the internal structure of arguments, and arguments are compared in terms of *specifity*. The second category includes systems in which preferences among arguments are derived from a priority relation on the rules in the underlying theory (e.g. [17,18]). In *Value Based Argumentation Frameworks*, the preference ordering on the set of arguments is derived from a preference ordering over the values that they promote (e.g. [19,20]). Finally, the abstract argumentation frameworks proposed by Amgoud and her colleagues ([21,22]) assume that preferences among arguments are induced by a preference relation defined on the underlying belief base.

Our argumentation framework is an extension of the framework of Governatori *et al.* [18], which is based on the grounded semantics of Dung's abstract argumentation framework [23] to provide an argumentative characterization of Defeasible Logic. In our framework, preferences are derived both from the structure of arguments - arguments that use local rules are considered stronger than those that use mapping rules - and from a preference ordering on the information sources (contexts). Our approach also shares common ideas with [21], which first introduced the notion of *contextual* preferences (in the form of several pre-orderings on the belief base), to take into account preferences that depend upon a particular context. The main differences are that in our case, these orderings are applied to the contexts themselves rather than directly to a set of arguments, and that we use a distributed underlying knowledge base.

3 Representation Model

We model a Multi-Context System C as a collection of distributed context theories C_i: A context is defined as a tuple of the form (V_i, R_i, T_i), where V_i is the vocabulary used by C_i, R_i is a set of rules, and T_i is a preference ordering on C.

V_i is a set of positive and negative literals. If q_i is a literal in V_i, $\sim q_i$ denotes the complementary literal, which is also in V_i. If q_i is a positive literal p then $\sim q_i$ is $\neg p$; and if q_i is $\neg p$, then $\sim q_i$ is p. We assume that each context uses a distinct vocabulary.

R_i consists of two sets of rules: the set of local rules and the set of mapping rules. The body of a local rule is a conjunction of *local* literals (literals that are contained in V_i), while its head contains a local literal:

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

Local rules express local knowledge and are interpreted in the classical sense: whenever the literals in the body of the rule are consequences of the local theory, then so is the literal in the head of the rule. Local rules with empty body denote factual knowledge.

Mapping rules associate local literals with literals from the vocabularies of other contexts (*foreign literals*). The body of each such rule is a conjunction of local and foreign literals, while its head contains a single local literal:

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

r_i^m associates local literals of C_i (e.g. a_i^1) with local literals of C_j (a_j^2), C_k (a_k^{n-1}) and possibly other contexts. a_i^n is a local literal of the theory that has defined r_i^m (C_i).

Finally, each context C_i defines a total preference ordering T_i on C to express its confidence in the knowledge it imports from other contexts. This is of the form:

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

According to T_i, C_k is preferred by C_i to C_l if C_k precedes C_l in T_i. The total preference ordering enables resolving all potential conflicts that may arise from the interaction of contexts through their mapping rules.

Example. Consider the following scenario. Dr. Amber has configured his mobile phone to decide whether it should ring based on his preferences and context. He has the following preferences: His mobile phone should ring in case of an incoming call (in_call) if it is in normal mode ($normal$) and he is not giving a course lecture ($lecture$). Dr. Amber is currently located in 'RA201' university classroom. It is class time, but he has just finished with a lecture and remains in the classroom reading his emails on his laptop. The mobile phone receives an incoming call, and it is in normal mode. The local knowledge of the mobile phone (C_1), which includes information about the mode of the phone and incoming calls, is encoded in the following local rules.

$r_{11}^l : in\_call_1, normal_1, \neg lecture_1 \rightarrow ring_1$
$r_{12}^l :\rightarrow in\_call_1$
$r_{13}^l :\rightarrow normal_1$

In case the mobile phone cannot reach a decision based on its local knowledge, it imports knowledge from other ambient agents. In this case, to determine whether Dr. Amber is giving a lecture, it connects through the university wireless network with Dr. Amber's laptop (C_2), a localization service (C_3) and the classroom manager (C_4, a stationary computer installed in 'RA201'), and imports information about Dr. Amber's scheduled events, location, and the classroom state through mapping rules r_{14}^m and r_{15}^m.

$r_{14}^m : classtime_2, location\_RA201_3 \Rightarrow lecture_1$
$r_{15}^m : \neg class\_activity_4 \Rightarrow \neg lecture_1$

The local context knowledge of the laptop, the localization service, and the classroom manager is expressed in rules r_{21}^l, r_{31}^l and $r_{41}^l - r_{42}^l$ respectively. The classroom manager infers whether there is active class activity, based on the number of people detected in the classroom ($detected$) by a person detection service.

$r_{21}^l :\rightarrow classtime_2$
$r_{31}^l :\rightarrow location\_RA201_3$
$r_{41}^l :\rightarrow detected(1)_4$
$r_{42}^l : detected(1)_4 \rightarrow \neg class\_activity_4$

The mobile phone is configured to give highest priority to information imported by the classroom manager and lowest priority to information imported by the laptop. This is encoded in preference ordering $T_1 = [C_4, C_3, C_2]$.

This example characterizes the type of applications, in which each ambient agent is aware of the type of knowledge that each of the other agents that it communicates with possesses, and has predefined how part of this knowledge relates to its local knowledge.

4 Argumentation Semantics

The argumentation framework that we propose uses arguments of local range, in the sense that each one is made of rules derived from a single context. Arguments made by different contexts are interrelated in the *Support Relation* through mapping rules. The Support Relation contains triples that represent proof trees for literals in the system. Each proof tree is made of rules of the context that the literal in its root is defined by. In case a proof tree contains mapping rules, for the respective triple to be contained in the Support Relation, similar triples for the foreign literals in the proof tree must have already been obtained. We should also note that, for sake of simplicity, we assume that there are no loops in the local context theories, and thus proof trees are finite. Loops in the local knowledge bases can be easily detected and removed without needing to interact with other agents. However, even if there are no loops in the local theories, the global knowledge base may contain loops caused by mapping rules.

Let $C = \{C_i\}$ be a MCS. The *Support Relation* of C (SR_C) is the set of all triples of the form (C_i, PT_{p_i}, p_i), where $C_i \in C$, $p_i \in V_i$, and PT_{p_i} is the proof tree for p_i based on the set of local and mapping rules of C_i. PT_{p_i} is a tree with nodes labeled by literals such that the root is labeled by p_i, and for every node with label q:

1. If $q \in V_i$ and $a_1, ..., a_n$ label the children of q then
   - If $\forall a_i \in \{a_1, ..., a_n\}$: $a_i \in V_i$ then there is a local rule $r_i \in C_i$ with body $a_1, ..., a_n$ and head q

- If $\exists a_j \in \{a_1, ..., a_n\}$ such that $a_j \notin V_i$ then there is a mapping rule $r_i \in C_i$ with body $a_1, ..., a_n$ and head q

2. If $q \in V_j \neq V_i$, then this is a leaf node of the tree and there is a triple of the form (C_j, PT_q, q) in SR_C
3. The arcs in a proof tree are labeled by the rules used to obtain them.

An *argument* A for a literal p_i is a triple (C_i, PT_{p_i}, p_i) in SR_C. Any literal labeling a node of PT_{p_i} is called *a conclusion* of A. However, when we refer to *the conclusion* of A, we refer to the literal labeling the root of PT_{p_i} (p_i). We write $r \in A$ to denote that rule r is used in the proof tree of A. A (proper) subargument of A is every argument with a proof tree that is (proper) subtree of the proof tree of A.

Based on the literals used in their proof trees, arguments are classified to *local* and *mapping* arguments. An argument A with conclusion $p_i \in V_i$ is a local argument of C_i if its proof tree contains only local literals of C_i (literals that are contained in V_i). Otherwise, A is a mapping argument of C_i. We denote as $Args_{C_i}$ the set of all arguments of C_i, while $Args_C$ is the set of all arguments in C: $Args_C = \bigcup Args_{C_i}$.

The derivation of local logical consequences in C_i is based on its local arguments. Actually, the conclusions of all local arguments in $Args_{C_i}$ are logical consequences of C_i. Distributed logical consequences are derived from a combination of local and mapping arguments in $Args_C$. In this case, we should also consider conflicts between competing rules, which are modeled as *attacks* between arguments, and preference orderings, which are used in our framework to *rank* mapping arguments.

The *rank of a literal* p in context C_i (denoted as $R(p, C_i)$) equals 0 if $p \in V_i$. If $p \in V_j \neq V_i$, then $R(p, C_i)$ equals the rank of C_j in T_i. The *rank of an argument* A in C_i (denoted as $R(A, C_i)$) equals the maximum between the ranks in C_i of the literals contained in A. It is obvious that for any three arguments A_1, A_2, A_3: If $R(A_1, C_i) \leq R(A_2, C_i)$ and $R(A_2, C_i) \leq R(A_3, C_i)$, then $R(A_1, C_i) \leq R(A_3, C_i)$; namely the preference relation $<$ on $Args^C$, which is build according to ordering T_i, is transitive.

The definitions of *attack* and *defeat* apply only for mapping arguments. An argument A *attacks* a mapping argument B at p_i, if p_i is a conclusion of B, $\sim p_i$ is a conclusion of A, and the subargument of B with conclusion p_i is not a local argument. An argument A *defeats* an argument B at p_i, if A attacks B at p_i, and for the subarguments of A, A' with conclusion $\sim p_i$, and of B, B' with conclusion p_i: $R(A', C_i) \leq R(B', C_i)$.

To link arguments through the mapping rules that they contain, we introduce in our framework the notion of *argumentation line*. An *argumentation line* A_L for a literal p_i is a sequence of arguments in $Args_C$, constructed in steps as follows:

- In the first step add in A_L one argument for p_i.
- In each next step, for each distinct literal q_j labeling a leaf node of the proof trees of the arguments added in the previous step, add one argument with conclusion q_j; the addition should not violate the following restriction.
- An argument B with conclusion q_j can be added in A_L only if A_L does not already contain a different argument D with conclusion q_j.

The argument for p_i added in the first step is called the *head argument* of A_L. If the number of steps required to build A_L is finite, then A_L is a finite argumentation line.

Infinite argumentation lines imply loops in the global knowledge base. Arguments contained in infinite lines participate in *attacks* against counter-arguments but may not be used to support the conclusion of their argumentation lines.

The notion of *supported* argument is meant to indicate when an argument may have an active role in proving or preventing the derivation of a conclusion. An argument A is *supported* by a set of arguments S if: (a) every proper subargument of A is in S; and (b) there is a finite argumentation line A_L with head A, such that every argument in $A_L - \{A\}$ is in S.

A mapping argument A is *undercut* by a set of arguments S if for every argumentation line A_L with head A, there is an argument B, such that B is supported by S, and B defeats a proper subargument of A or an argument in $A_L - \{A\}$. That an argument A is *undercut* by a set of arguments S means that we can show that some premises of A cannot be proved if we accept the arguments in S.

An argument A is *acceptable* w.r.t a set of arguments S if:

1. A is a local argument; or
2. A is supported by S and every argument defeating A is undercut by S

Intuitively, that an argument A is *acceptable* w.r.t. S means that if we accept the arguments in S as valid arguments, then we feel compelled to accept A as valid. Based on the concept of acceptable arguments, we define *justified arguments* and *justified literals*. J_i^C is defined as follows:

- $J_0^C = \emptyset$;
- $J_{i+1}^C = \{A \in Args_C \mid A \text{ is acceptable w.r.t. } J_i^C\}$

The set of *justified arguments* in a MCS C is $JArgs^C = \bigcup_{i=1}^{\infty} J_i^C$. A literal p_i is *justified* if it is the conclusion of an argument in $JArgs^C$. That an argument A is justified means that it resists every reasonable refutation. That a literal p_i is justified, it actually means that it is a logical consequence of C.

Finally, we also introduce the notion of *rejected arguments* and *rejected literals* for the characterization of conclusions that do not derive from C. An argument A is *rejected* by sets of arguments S, T when:

1. A is not a local argument, and either
2. (a) a proper subargument of A is in S; or
   (b) A is defeated by an argument supported by T; or
   (c) for every argumentation line A_L with head A there exists an argument $A' \in A_L - \{A\}$, such that either a subargument of A' is in S; or A' is defeated by an argument supported by T

That an argument is rejected by sets of arguments S and T means that either it is supported by arguments in S, which can be thought of as the set of already rejected arguments, or it cannot overcome an attack from an argument supported by T, which can be thought of as the set of justified arguments. Based on the definition of rejected arguments, we define R_i^C as follows:

- $R_0^C = \emptyset$;
- $R_{i+1}^C = \{A \in Args_C \mid A \text{ is rejected by } R_i^C, JArgs^C\}$

The set of *rejected arguments* in a MCS C is $RArgs^C = \bigcup_{i=1}^{\infty} R_i^C$. A literal p_i is *rejected* if there is no argument in $Args^C - RArgs^C$ with conclusion p_i. That a literal is rejected means that we are able to prove that it is not a logical consequence of C.

Example (continued). Given the MCS C of the example, in_call_1, $normal_1$, $classtime_2$, $location\_RA201_3$, $detected(1)_4$, and $\neg class\_activity_4$ are justified in C, since they are supported by local arguments; all these arguments are in J_i^C for every i. The argument A=$\{\neg class\_activity_4 \Rightarrow \neg lecture_1\}$ is supported by J_1^C, as there is an argument for $\neg class\_activity_4$ in J_1^C. Moreover, it is not defeated by attacking argument B=$\{classtime_2, location\_RA201_3 \Rightarrow lecture_1\}$, as B has higher rank than A in C_1. Hence, A is in J_2, and $\neg lecture_1$ is justified in C. Argument D for $ring_1$, which is derived from the arguments for in_call_1, $normal_1$ and $\neg lecture_1$ (A) and rule r_{11}^l, is supported by J_2^C and not defeated by attacking argument B. Therefore, D is justified, and $ring_1$ is justified in C.

Lemmata 1-3 describe some formal properties of the framework. Their proofs are available at: www.csd.uoc.gr/∼bikakis/thesis.pdf. Lemma 1 refers to the monotonicity in J_i^C and $R_i^C(T)$, while Lemma 2 represents the fact that no argument is both "believed" and "disbelieved".

Lemma 1. *The sequences J_i^C and $R_i^C(T)$ are monotonically increasing.*

Lemma 2. *In a Multi-Context System C, no literal is both justified and rejected.*

If consistency is assumed in the local rules of a context theory (two complementary conclusions may not be derived as local consequences of a context theory), then using Lemma 2, it is easy to prove that the entire framework is consistent (Lemma 3).

Lemma 3. *If the set of justified arguments in C, $JArgs^C$, contains two arguments with complementary conclusions, then both are local arguments of the same context.*

5 Distributed Query Evaluation

$P2P\_DR$ is a distributed algorithm for query evaluation that implements the proposed argumentation framework. The specific problem that it deals with is: *Given a MCS C, and a query about literal p_i issued to context C_i, compute the truth value of p_i*. For an arbitrary literal p_i, $P2P\_DR$ returns one of the following values: (a) *true*; indicating that p_i is justified in C; (b) *false*; indicating that p_i is rejected in C; or (c) *undefined*; indicating that p_i is neither justified nor rejected in C.

5.1 Algorithm Description

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

In step 2 (lines 9-12), $P2P\_DR$ calls $Support$ (described later in this section) to determine whether there are *applicable* and *unblocked* rules with head p_i. We call *applicable* those rules that for all literals in their body $P2P\_DR$ has computed *true* as their

truth value, while *unblocked* are the rules that for all literals in their body $P2P\_DR$ has computed either *true* or *undefined* as their truth values. $Support$ returns two data structures for p_i: (a) the Supportive Set of p_i (SS_{p_i}), which is the set of foreign literals used in the most preferred (according to T_i) chain of applicable rules for p_i ; and (b) the Blocking Set of p_i (BS_{p_i}), which is the set of foreign literals used in the most preferred chain of unblocked rules for p_i. If there is no unblocked rule for p_i ($BS_{p_i} = \emptyset$), $P2P\_DR$ returns *false* as an answer and terminates. Similarly, in step 3 (lines 13-14), $P2P\_DR$ calls $Support$ to compute the respective constructs for $\sim p_i$ ($SS_{\sim p_i}$,$BS_{\sim p_i}$).

In the last step (lines 15-24), $P2P\_DR$ uses the constructs computed in the previous steps and preference ordering T_i to compute the answer for p_i. In case there is no unblocked rule for $\sim p_i$ ($BS_{\sim p_i} = \emptyset$), or SS_{p_i} is computed by $Stronger$ (described later in this section) to be *stronger* than $BS_{\sim p_i}$, $P2P\_DR$ returns Ans_{p_i}=*true*. That SS_{p_i} is *stronger* than BS_{p_i} means that the chains of applicable rules for p_i involve information from more preferred contexts to those that are involved in the chains of unblocked rules for $\sim p_i$. If there is at least one applicable rule for $\sim p_i$, and BS_{p_i} is *not stronger* than $SS_{\sim p_i}$, $P2P\_DR$ returns *false*. In any other case, it returns *undefined*.

The context that is called to evaluate the query for p_i (C_i) returns through Ans_{p_i} the truth value for p_i. SS_{p_i} and BS_{p_i} are returned to the querying context (C_0) only if the two contexts (C_0 and C_i) are actually the same context. Otherwise, the empty set is assigned to both SS_{p_i} and BS_{p_i} and returned to C_0. In this way, the size of the messages exchanged between different contexts is kept small. $Hist_{p_i}$ is a structure used by $Support$ to detect loops in the global knowledge base. The algorithm parameters are:

- p_i: the queried literal (input)
- C_0: the context that issues the query (input)
- C_i: the context that defines p_i (input)
- $Hist_{p_i}$: the list of pending queries ($[p_1, ..., p_i]$) (input)
- T_i: the preference ordering of C_i (input)
- SS_{p_i}: a set of foreign literals of C_i denoting the Supportive Set of p_i (output)
- BS_{p_i}: a set of foreign literals of C_i denoting the Blocking Set of p_i (output)
- Ans_{p_i}: the answer returned for p_i (output)

P2P_DR($p_i, C_0, C_i, Hist_{p_i}, T_i, SS_{p_i}, BS_{p_i}, Ans_{p_i}$)

1: call $local\_alg(p_i, localAns_{p_i})$
2: **if** $localAns_{p_i} = true$ **then**
3: $\quad Ans_{p_i} \leftarrow true$, $SS_{p_i} \leftarrow \emptyset$, $BS_{p_i} \leftarrow \emptyset$
4: $\quad$ terminate
5: call $local\_alg(\sim p_i, localAns_{\sim p_i})$
6: **if** $localAns_{\sim p_i} = true$ **then**
7: $\quad Ans_{p_i} \leftarrow false$, $SS_{p_i} \leftarrow \emptyset$, $BS_{p_i} \leftarrow \emptyset$
8: $\quad$ terminate
9: call $Support(p_i, Hist_{p_i}, T_i, SS_{p_i}, BS_{p_i})$
10: **if** $BS_{p_i} = \emptyset$ **then**
11: $\quad Ans_{p_i} \leftarrow false$, $SS_{p_i} \leftarrow \emptyset$, $BS_{p_i} \leftarrow \emptyset$
12: $\quad$ terminate
13: $Hist_{\sim p_i} \leftarrow (Hist_{p_i} - \{p_i\}) \cup \{\sim p_i\}$
14: call $Support(\sim p_i, Hist_{\sim p_i}, T_i, SS_{\sim p_i}, BS_{\sim p_i})$
15: **if** $SS_{p_i} \neq \emptyset$ and ($BS\sim p_i = \emptyset$ or $Stronger(SS_{p_i}, BS_{\sim p_i}, T_i) = SS_{p_i}$) **then**

16: $Ans_{p_i} \leftarrow true$
17: **if** $C_0 \neq C_i$ **then**
18: $SS_{p_i} \leftarrow \emptyset$, $BS_{p_i} \leftarrow \emptyset$
19: **else if** $SS_{\sim p_i} \neq \emptyset$ and $Stronger(BS_{p_i}, SS_{\sim p_i}, T_i) \neq BS_{p_i}$ **then**
20: $Ans_{p_i} \leftarrow false$, $SS_{p_i} \leftarrow \emptyset$, $BS_{p_i} \leftarrow \emptyset$
21: **else**
22: $Ans_{p_i} \leftarrow undefined$
23: **if** $C_0 \neq C_i$ **then**
24: $SS_{p_i} \leftarrow \emptyset$, $BS_{p_i} \leftarrow \emptyset$

$local\_alg$ is called by $P2P\_DR$ to determine whether the truth value of the queried literal can be derived from the local rules of a context.

local_alg($p_i, localAns_{p_i}$)
1: **for all** $r_i \in R^s[p_i]$ **do**
2: **for all** $b_i \in body(r_i)$ **do**
3: call $local\_alg(b_i, localAns_{b_i})$
4: **if** for all b_i: $localAns_{b_i} = true$ **then**
5: $localAns_{p_i} \leftarrow true$
6: terminate
7: $localAns_{p_i} \leftarrow false$

$Support$ is called by $P2P\_DR$ to compute SS_{p_i} and BS_{p_i}. To compute these structures, it checks the applicability of the rules with head p_i, using the truth values of the literals in their body, as these are evaluated by $P2P\_DR$. To avoid loops, before calling $P2P\_DR$, it checks if the same query has been issued before during the running call of $P2P\_DR$. For each applicable rule r_i, $Support$ builds its Supportive Set, SS_{r_i}; this is the union of the set of *foreign literals* contained in the body of r_i with the Supportive Sets of the local literals contained in the body of the rule. Similarly, for each unblocked rule r_i, it computes its Blocking Set BS_{r_i} using the Blocking Sets of its body literals. $Support$ computes the Supportive Set of p_i, SS_{p_i}, as the *strongest* rule Supportive Set SS_{r_i}; and its Blocking Set, BS_{p_i}, as the *strongest* rule Blocking Set BS_{r_i}, using the $Stronger$ function. The parameters of $Support$ are:

- p_i: the queried literal (input)
- $Hist_{p_i}$: the list of pending queries ($[p_1, ..., p_i]$) (input)
- T_i: the preference ordering of C_i (input)
- SS_{p_i}: the Supportive Set of p_i (output)
- BS_{p_i}: the Blocking Set of p_i (output)

Support($p_i, Hist_{p_i}, T_i, SS_{p_i}, BS_{p_i}$)
1: **for all** $r_i \in R[p_i]$ **do**
2: $cycle(r_i) \leftarrow false$
3: $SS_{r_i} \leftarrow \emptyset$, $BS_{r_i} \leftarrow \emptyset$
4: **for all** $b_t \in body(r_i)$ **do**
5: **if** $b_t \in Hist_{p_i}$ **then**
6: $cycle(r_i) \leftarrow true$
7: $BS_{r_i} \leftarrow BS_{r_i} \cup \{d_t\}$ {$d_t \equiv b_t$ if $b_t \notin V_i$; otherwise d_t is the first foreign literal of C_i added in $Hist_{p_i}$ after b_t}

else
  $Hist_{b_t} \leftarrow Hist_{p_i} \cup \{b_t\}$
  call $P2P\_DR(b_t, C_i, C_t, Hist_{b_t}, T_t, SS_{b_t}, BS_{b_t}, Ans_{b_t})$
  if $Ans_{b_t} = false$ **then**
    stop and check the next rule
  else if $Ans_{b_t} = undefined$ or $cycle(r_i) = true$ **then**
    $cycle(r_i) \leftarrow true$
    if $b_t \notin V_i$ **then**
      $BS_{r_i} \leftarrow BS_{r_i} \cup \{b_t\}$
    else
      $BS_{r_i} \leftarrow BS_{r_i} \cup BS_{b_t}$
  else
    if $b_t \notin V_i$ **then**
      $BS_{r_i} \leftarrow BS_{r_i} \cup \{b_t\}$
      $SS_{r_i} \leftarrow SS_{r_i} \cup \{b_t\}$
    else
      $BS_{r_i} \leftarrow BS_{r_i} \cup BS_{b_t}$
      $SS_{r_i} \leftarrow SS_{r_i} \cup SS_{b_t}$
if $BS_{p_i} = \emptyset$ or $Stronger(BS_{r_i}, BS_{p_i}, T_i) = BS_{r_i}$ **then**
  $BS_{p_i} \leftarrow BS_{r_i}$
if $cycle(r_i) = false$ **then**
  if $SS_{p_i} = \emptyset$ or $Stronger(SS_{r_i}, SS_{p_i}, T_i) = SS_{r_i}$ **then**
    $SS_{p_i} \leftarrow SS_{r_i}$

$Stronger(A, B, T_i)$ returns the *strongest* between two sets of literals, A and B, according to preference ordering T_i. A literal a_k is *preferred* to literal b_j, if C_k precedes C_l in T_i. The strength of a set is determined by the the least preferred literal in this set.

Stronger(A, B, T_i)
if $\exists b_j \in B$: $\forall a_k \in A$: C_k has lower rank than C_j in T_i **then**
  $Stronger = A$
else if $\exists a_k \in A$: $\forall b_j \in B$: C_j has lower rank than C_k in T_i **then**
  $Stronger = B$
else
  $Stronger = None$

Example (continued). Given a query about $ring_1$, $P2P\_DR$ proceeds as follows. It fails to compute an answer based on C_1's local theory, and uses rules r^m_{14} and r^m_{15} to compute an answer for $\neg lecture_1$. Using the local rules of C_2, C_3 and C_4, it computes positive answers for $classtime_2$, $location\_RA201_3$ and $\neg class\_activity_4$ respectively, determines that both r^m_{14} and r^m_{15} are applicable, and computes their Supportive Sets: $SS_{r^m_{14}} = \{class_2, location\_RA201_3\}$ and $SS_{r^m_{15}} = \{\neg class\_activity_4\}$. As C_4 precedes C_2 in T_1, $P2P\_DR$ determines that $SS_{r^m_{15}}$ is stronger, computes a positive answer for $\neg lecture_1$, and eventually (using rule r^l_{11}) returns a positive answer (*true*) for $ring_1$.

5.2 Properties of the Algorithm

Below, we describe formal properties of $P2P\_DR$ regarding its termination, soundness and completeness w.r.t. the argumentation framework and complexity. The proofs for

the following propositions are available at www.csd.uoc.gr/∼bikakis/thesis.pdf. Proposition 1 is a consequence of the cycle detection process within the algorithm.

Proposition 1. *The algorithm is guaranteed to terminate returning one of the values true, false and undefined as an answer for the queried literal.*

Proposition 2 associates the answers produced by $P2P\_DR$ with the concepts of justified and rejected literals.

Proposition 2. *For a Multi-Context System C and a literal p_i in C, $P2P\_DR$ returns:*

1. $Ans_{p_i} = true$ *iff* p_i *is justified in* C
2. $Ans_{p_i} = false$ *iff* p_i *is rejected in* C
3. $Ans_{p_i} = undefined$ *iff* p_i *is neither justified nor rejected in* C

Propositions 3 and 4 are consequences of two states that we retain for each context, which keep track of the results for all incoming and outgoing queries. The worst case that both propositions refer to is when all rules of C_i contain either p_i (the queried literal) or $\sim p_i$ in their head and all system literals in their bodies.

Proposition 3. *The total number of messages exchanged between the system contexts for the evaluation of a query is, in the worst case, $O(n \times \sum P(n,k))$, where n stands for the total number of literals in the system, $\sum$ expresses the sum over $k = 0, 1, ..., n$, and $P(n,k)$ stands for the number of permutations with length k of n elements. In case, there are no loops in the global knowledge base, the number of messages is polynomial to the size of the global knowledge base.*

Proposition 4. *The number of operations imposed by one call of $P2P\_DR$ for the evaluation of a query for literal p_i is, in the worst case, proportional to the number of rules in C_i, and to the total number of literals in the system.*

6 Conclusion

This paper proposes a totally distributed approach for contextual reasoning in Ambient Intelligence, based on representing context knowledge of ambient agents as context theories in a Multi-Context System, and reasoning with the available knowledge using arguments. Using a total preference ordering on the system contexts, our approach enables resolving all conflicts that arise from the interaction of contexts through their mappings. The paper also presents a distributed algorithm for query evaluation that implements the proposed argumentation framework, and studies its formal properties.

Our ongoing work involves: (a) studying alternative methods for conflict resolution, which differ in the way that agents evaluate the imported context information; (b) adding non-monotonic features to the local context theories to support uncertainty in the local context knowledge; (c) extending our approach 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 Ambient Intelligence environments.

References

1. Henricksen, K., Indulska, J.: Modelling and Using Imperfect Context Information. In: Proceedings of PERCOMW 2004, Washington, DC, USA, pp. 33–37. IEEE Computer Society, Los Alamitos (2004)
2. Bikakis, A., Patkos, T., Antoniou, G., Plexousakis, D.: A Survey of Semantics-based Approaches for Context Reasoning in Ambient Intelligence. In: Constructing Ambient Intelligence. Communications in Computer and Information Science, pp. 14–23. Springer, Heidelberg (2008)
3. Bikakis, A., Antoniou, G., Hassapis, P.: Alternative Strategies for Conflict Resolution in Multi-Context Systems. In: Proceedings of the 5th IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI 2009). Springer, Heidelberg (2009)
4. McCarthy, J.: Generality in Artificial Intelligence. Communications of the ACM 30(12), 1030–1035 (1987)
5. Buvac, S., Mason, I.A.: Propositional Logic of Context. In: AAAI, pp. 412–419 (1993)
6. Giunchiglia, F., Serafini, L.: Multilanguage hierarchical logics, or: how we can do without modal logics. Artificial Intelligence 65(1) (1994)
7. Ghidini, C., Giunchiglia, F.: Local Models Semantics, or contextual reasoning=locality+compatibility. Artificial Intelligence 127(2), 221–259 (2001)
8. Serafini, L., Bouquet, P.: Comparing formal theories of context in AI. Artificial Intelligence 155(1-2), 41–67 (2004)
9. Roelofsen, F., Serafini, L.: Minimal and Absent Information in Contexts. In: IJCAI, pp. 558–563 (2005)
10. Brewka, G., Roelofsen, F., Serafini, L.: Contextual Default Reasoning. In: IJCAI, pp. 268–273 (2007)
11. Sabater, J., Sierra, C., Parsons, S., Jennings, N.R.: Engineering Executable Agents using Multi-context Systems. Journal of Logic and Computation 12(3), 413–442 (2002)
12. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Inconsistency tolerance in P2P data integration: An epistemic logic approach. In: Bierman, G., Koch, C. (eds.) DBPL 2005. LNCS, vol. 3774, pp. 90–105. Springer, Heidelberg (2005)
13. Chatalic, P., Nguyen, G.H., Rousset, M.C.: Reasoning with Inconsistencies in Propositional Peer-to-Peer Inference Systems. In: ECAI, pp. 352–356 (2006)
14. Binas, A., Sheila, A.: Peer-to-Peer Query Answering with Inconsistent Knowledge. In: KR, pp. 329–339 (2008)
15. Simari, G.R., Loui, R.P.: A Mathematical Treatment of Defeasible Reasoning and its Implementation. Artificial Intelligence 53(2-3), 125–157 (1992)
16. Stolzenburg, F., García, A.J., Chesñevar, C.I., Simari, G.R.: Computing Generalized Specificity. Journal of Applied Non-Classical Logics 13(1), 87–113 (2003)
17. Prakken, H., Sartor, G.: Argument-Based Extended Logic Programming with Defeasible Priorities. Journal of Applied Non-Classical Logics 7(1) (1997)
18. Governatori, G., Maher, M.J., Billington, D., Antoniou, G.: Argumentation Semantics for Defeasible Logics. Journal of Logic and Computation 14(5), 675–702 (2004)
19. Bench-Capon, T.: Persuasion in Practical Argument Using Value-based Argumentation Frameworks. Journal of Logic and Computation 13, 429–448 (2003)
20. Kaci, S., van der Torre, L.: Preference-based argumentation: Arguments supporting multiple values. Internation Journal of Approximate Reasoning 48(3), 730–751 (2008)

21. Amgoud, L., Parsons, S., Perrussel, L.: An Argumentation Framework based on contextual Preferences. In: International Conference on Formal and Applied and Practical Reasoning (FAPR 2000), pp. 59–67 (2000)
22. Amgoud, L., Cayrol, C.: A Reasoning Model Based on the Production of Acceptable Arguments. Annals of Mathematic and Artificial Intelligence 34(1-3), 197–215 (2002)
23. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. 77, 321–357 (1995)