Solving Power and Trust Conflicts through Argumentation in Agent-mediated Knowledge Distribution*

Carlos I. Chesñevar[1], Ramón Brena[2] and José L. Aguirre[2]

[1]Artificial Intelligence Research Group – Universitat de Lleida
25001 Lleida, Spain – cic@eps.udl.es

[2]Centro de Sistemas Inteligentes – Tecnológico de Monterrey
64849 Monterrey, N.L., México – {ramon.brena, jlaguirre}@itesm.mx

Abstract

Distributing pieces of knowledge in large, usually distributed organizations is a central problem in Knowledge and Organization management. Policies for distributing knowledge and information are mostly incomplete or in potential conflict with each other. As a consequence, decision processes for information distribution may be difficult to formalize on the basis of a rationally justified procedure. This article presents an argumentative approach to cope with this problem based on integrating the JITIK multiagent system with Defeasible Logic Programming (DeLP), a logic programming formalism for defeasible argumentation. We show how power relations, as well as delegation and trust, can be embedded within our framework in terms of DeLP, in such a way that a dialectical argumentation process works as a decision core. Conflicts among policies are solved on the basis of a dialectical analysis whose outcome determines to which specific users different pieces of knowledge are to be delivered.

Key words: Defeasible Argumentation, Knowledge Management, Multiagent systems

*This article is an extended version of preliminary research work presented in [12].

1 Introduction and Motivations

Information and knowledge (IK) have been identified as valuable assets [8, 26] in modern organization nowadays, motivating the development of different Knowledge Management (KM) techniques. In the context of KM, distributing customized pieces of IK in large and distributed organizations is a central process. This process turns out to be a decision making problem as well, because IK to be distributed to different users is not the same, so it must be decided upon which IK item goes to each user. Organization policies for IK dissemination should be defined to deliver notifications to specific organization members, according to management criteria. Indeed, a central concern in KM is to facilitate *knowledge flow* within relevant actors within an organization. Organizations typically have different criteria establishing their information distribution *policies*, and in many real situations these policies conflict with each other.

The influence of *power relations* on policies for information distribution inside organizations has been the subject of many studies in Knowledge Management (KM) [27, 22]. Even though modern organization theories emphasize flexibility and learning over rigid hierarchical structures [26], formal power relations have remained as a key component in any large organization. A counterpart of formal relations are informal relations in organizations, notably *trust* relations [19], which are normally not represented in the organization's formal structure, but are nonetheless extremely important. Trust and reputation relations have been formalized for computational purposes [34, 30] with the goal of defining what is believed to be reliable in the context of such informal relations.

A common feature characterizing these policies is the presence of defeasibility, i.e. policies may change in the light of new information, such as particular interests and/or information needs of users, exceptional situations, etc. Thus, making decisions about whether to deliver or not a specific piece of information to certain users is indeed a challenging problem, particularly in the context of Agent-mediated Knowledge Management. In this context, *defeasible argumentation* [13, 32] has evolved in the last decade as a successful multi-disciplinary approach to formalize commonsense reasoning and decision making problems as the ones discussed before.

This article presents a novel approach to solve the above problem in knowledge distribution in large organizations, based on integrating JITIK (a multiagent Knowledge Management system) with Defeasible Logic Programming (DeLP), a logic programming formalism for defeasible argumentation. We show how power relations, as well as delegation and trust, can be embedded within our framework in terms of DeLP, in such a way that a dialectical argumentation process works as a decision core. We show how to represent power and trust capabilities associated with the agents involved, encompassing both formal and informal relations in organizations. Conflicts emerging from potentially contradictory policies as well as from trust and empowerment issues are solved on the basis of a dialectical analysis whose outcome determines whether a particular information item should be delivered or not to a specific user.

2 The JITIK Framework

JITIK, which stands for "Just-In-Time Information and Knowledge" [4, 5, 1], is a multiagent-based system for disseminating pieces of IK among the members of a large or distributed organization, thus supporting a Knowledge-management function. It is aimed to deliver the right IK to the adequate people just-in-time. The JITIK agent model is shown in Fig. 1. *Personal Agents* work on behalf of the members of the organization. They filter and deliver useful content according to user preferences. The Site Agent provides of IK to the Personal Agents, acting as a broker between them and Service agents. *Service agents* collect and detect IK pieces that are supposed to be relevant for someone in the organization. Examples of service agents are the Web Service agents, which receive and process external requests, as well as monitor agents which are continuously monitoring sources of IK (web pages, databases, etc.). Other Service agents monitor at time intervals the state of an IK resource, like a web page, data in an enterprise's database, etc.

The *Ontology agent* contains knowledge about the interest areas for the members of the organization and about its structure [6]. We store this information in *ontologies* [3]. Ontologies are structured representation of concepts, classes, individuals, properties, and other categories. We used open standards like DAML-OIL[23], which allow to publish in the internet ontological knowledge in a way understandable both by humans and machines.

In JITIK ontologies we usually store relevant taxonomies, as the ones for interest areas, as well as the structure of the organization. For example, in an academic institution, the interest areas could be the science domains in which the institution is specialized, and the organizational chart of the

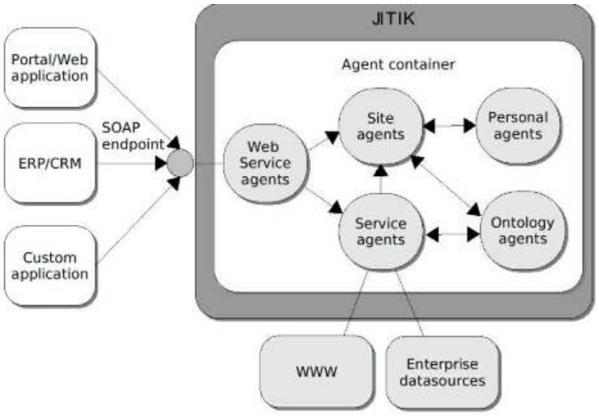

Figure 1: The JITIK agent model

institution gives the structure of the organization.

Site agents are the heart of a "cluster" composed by one site agent and several personal agents served by the former. In an organization, clusters would be associated to departments, divisions, etc., depending on the size of them. Networks can be made up connecting several site agents. Distributed organizations like multinational companies would have a web of many connected site agents. Among the services provided by JITIK we have the following:

Recommendation services : A user's profile is represented by a set of points in the taxonomies, as each user could have many interests and could be located at different parts of the organizational structure. As JITIK keeps track of user interests and preferences it is able to recommend content to users on demand. Recommended content may be used in Portals or Web applications.

Subscription services : JITIK allows users to subscribe to changes in specific areas. Also, users may customize the media and frequency of JITIK notifications using using simple web-based interfaces by means of rules. Rules may be defined so as messages relative to certain topics are handled with higher priorities. A rule may state that several alerts should be sent to their cell-phone via SMS, and also define that interest-area messages be sent in a weekly summary via email.

Content distribution services : Enterprise applications can deliver content to the system using its semantic-based content distribution services. When new content is received it is classified and distributed to those users who could be interested. Users receive the notifications of new content as specified by their own rules.

3 Modelling Conflicting Policies in JITIK: an argument-based approach

As explained before, JITIK aims at disseminating pieces of IK among the members of a large or distributed organization, thus supporting a Knowledge-management function. The Site Agent is in charge of providing IK to the Personal Agents, acting as a broker between them and Service agents. Clearly, in large or distributed organizations there are usually complex decision-making situations regarding IK distribution, specially in the presence of *potentially incomplete information* concerning metadata and user profiles, as well as *competing policies*, which can include several exceptions. Therefore, it is important that Site agents are provided with appropriate knowledge representation and inference capabilities to solve such problems.

Next we will provide a basic formalization for the main problem a JITIK Site Agent has to solve, namely *distributing items among users according to possibly conflicting policies.* Consider a set $I = \{i_1, i_2, \ldots, i_k\}$ of information items, which has to be distributed among a set $U = \{u_1, \ldots, u_s\}$ of users. Every item $i \in I$ should be delivered to only a distinguished subset of U. To accomplish this task, the Site Agent will apply a distribution policy p, which can be formally defined as a mapping $p : I \longrightarrow \wp(U)$. Distributing an item i to a user u is *compliant* with a policy p whenever $(i, \{\ldots, u, \ldots\}) \in p$.

In any real-world organization, it is clear that policies are not formulated in this way, but instead they are specified by a number of *constraints* enforced by the organization (e.g. access rights, power relationships, etc.). If P is a set of possible policies in the organization, given two policies $p_1, p_2 \in P$, we say they are in *conflict* whenever $(i, \{\ldots, u, \ldots\}) \in p_1$ but $(i, \{\ldots, u, \ldots\}) \not\in p_2$, or viceversa. A conflict means that an information item i cannot be compliant with two policies p_1 and p_2 at the same time. We can define a *dominance* partial order $\prec$ among possible policies in P, writing $p_1 \prec p_2$ to indicate that a policy p_2 is preferred over policy p_1 in case they are in conflict. In this setting, the "information distribution problem" to be solved by a JITIK Site Agent could then be recast as follows:

> *Send every information item $i \in I$ to a user $u \in U$ following a distribution policy p iff (a) i is compliant with p; and (b) p is preferred wrt to every non-dominated policy $p' \in P$.*[1]

[1] Note that characterizing p depends on the specific sets U, I and P under consideration.

Note that dominance characterizes a transitive ordering among policies, with the following particular feature: a policy p_i can be dominated by another policy p_j, making p_i not valid. However, p_j can on its turn be dominated by another policy p_k. If that is the case, the policy p_i may be perhaps valid again (as the policy p_j was "defeated" by policy p_k).

Conflicting situations like the ones described before can be nicely recast in argument-based reasoning systems [13]. In fact, during the last decade defeasible argumentation has emerged as a new reasoning paradigm, which provides a useful setting to formalize commonsense qualitative reasoning in a computationally attractive way. Recent research has shown that argumentation can be integrated in a growing number of real-world applications such as multiagent systems [31], knowledge engineering [7], and clustering [21], among many others.

Several defeasible argumentation frameworks have been developed on the basis of extensions to logic programming (see [13, 32]). *Defeasible logic programming*(DeLP) [18] is one of such formalisms, combining results from defeasible argumentation theory [35] and logic programming. DeLP is a multipurpose programming language, which combines powerful representation features inherited from logic programming as well as argumentative reasoning capabilities in a unified framework. DeLP has proven to be particularly attractive in the context of many real-world applications, such as clustering [21], natural language processing and recommender systems [14], among others. DeLP provides the possibility of representing information in the form of defeasible and strict rules in a declarative manner. An important characteristic of the DeLP approach is that by performing defeasible reasoning dialectically it can deal successfully with potentially contradictory information. The process of deciding if a conclusion C is supported, or *warranted*, begins by analyzing if there exists an undefeated argument (a *warrant*) supporting C, i.e, an argument for which every possible attacking argument has been defeated. An attack becomes a defeat when the attacking argument is better, in some specific sense, than the supporting argument. In order to make this article self-contained, in the next Section we will present a brief overview of the DeLP framework.

Here we do not discuss the problem of finding out whether such a mapping actually exists, but rather focus on enforcing dominance on conflicting policies.

4 Defeasible Logic Programming: overview

Defeasible logic programming (DeLP) [18] is a defeasible argumentation formalism based on logic programming. A defeasible logic program[2] is a set $K = (\Pi, \Delta)$ of Horn-like clauses, where Π and Δ stand for sets of strict and defeasible knowledge, respectively. The set Π of strict knowledge involves *strict rules* of the form $p \ \leftarrow \ q_1, \ldots, q_k$ and *facts* (strict rules with empty body), and it is assumed to be *non-contradictory.* The set Δ of defeasible knowledge involves *defeasible rules* of the form $p \;{-\!\!\prec}\; q_1, \ldots, q_k$, which stands for "$q_1, \ldots q_k$ provide a *tentative reason* to believe p." The underlying logical language is that of extended logic programming, enriched with a special symbol "${-\!\!\prec}$" to denote defeasible rules. Both default and classical negation are allowed (denoted not and $\sim$, resp.). Syntactically, the symbol "${-\!\!\prec}$" is all that distinguishes a *defeasible* rule $p \;{-\!\!\prec}\; q_1, \ldots q_k$ from a *strict* (non-defeasible) rule $p \ \leftarrow \ q_1, \ldots, q_k$. DeLP rules are thus Horn-like clauses to be thought of as *inference rules* rather than implications in the object language. Deriving literals in DeLP results in the construction of *arguments.*

Definition 1 (Argument) *Given a DeLP program $\mathcal{P}$, an* argument $\mathcal{A}$ *for a query q, denoted $\langle \mathcal{A}, q \rangle$, is a subset of ground instances of defeasible rules in $\mathcal{P}$ and a (possibly empty) set of default ground literals "$\mathsf{not}\ L$", such that: 1) there exists a* defeasible derivation *for q from $\Pi \cup \mathcal{A}$; 2) $\Pi \cup \mathcal{A}$ is non-contradictory (i.e, $\Pi \cup \mathcal{A}$ does not entail two complementary literals p and $\sim p$ (or p and $\mathsf{not}\ p$)), and 3) $\mathcal{A}$ is minimal with respect to set inclusion.*

An argument $\langle \mathcal{A}_1, Q_1 \rangle$ is a sub-argument *of another argument $\langle \mathcal{A}_2, Q_2 \rangle$ if $\mathcal{A}_1 \subseteq \mathcal{A}_2$. Given a DeLP program $\mathcal{P}$, $Args(\mathcal{P})$ denotes the set of all possible arguments that can be derived from $\mathcal{P}$.*

The notion of defeasible derivation corresponds to the usual query-driven SLD derivation used in logic programming, performed by backward chaining on both strict and defeasible rules; in this context a negated literal $\sim p$ is treated just as a new predicate name *no_p*. Minimality imposes a kind of 'Occam's razor principle' on arguments. The non-contradiction requirement forbids the use of (ground instances of) defeasible rules in an argument $\mathcal{A}$ whenever $\Pi \cup \ \mathcal{A}$ entails two complementary literals.

[2] When it is clear from the context we will simply refer to a defeasible logic program as a "DeLP program" or just "program"

Definition 2 (Counterargument – Defeat) *An argument $\langle \mathcal{A}_1, q_1\rangle$ is a* counterargument *for an argument $\langle \mathcal{A}_2, q_2\rangle$ iff (1) There is an subargument $\langle \mathcal{A}, q\rangle$ of $\langle \mathcal{A}_2, q_2\rangle$ such that the set $\Pi \cup \{q_1, q\}$ is contradictory; (2) A literal* not q_1 *is present in some rule in $\mathcal{A}_1$.A partial order $\preceq \subseteq$ Args($\mathcal{P}$) $\times$ Args($\mathcal{P}$) will be used as a* preference criterion[3] *among conflicting arguments. An argument $\langle \mathcal{A}_1, q_1\rangle$ is a* defeater *for an argument $\langle \mathcal{A}_2, q_2\rangle$ if $\langle \mathcal{A}_1, q_1\rangle$ counterargues $\langle \mathcal{A}_2, q_2\rangle$, and $\langle \mathcal{A}_1, q_1\rangle$ is preferred over $\langle \mathcal{A}_2, q_2\rangle$ wrt $\preceq$. For cases (1) and (2) above, we distinguish between* proper *and* blocking defeaters *as follows:*

- *In case (1) the argument $\langle \mathcal{A}_1, q_1\rangle$ will be called a* proper defeater *for $\langle \mathcal{A}_2, q_2\rangle$ iff $\langle \mathcal{A}_1, q_1\rangle$ is strictly preferred over $\langle \mathcal{A}, q\rangle$ wrt $\preceq$.*
- *In case (1), if $\langle \mathcal{A}_1, q_1\rangle$ and $\langle \mathcal{A}, q\rangle$ are unrelated to each other, or in case (2), $\langle \mathcal{A}_1, q_1\rangle$ will be called a* blocking defeater *for $\langle \mathcal{A}_2, q_2\rangle$.*

Specificity [35] is used in DeLP as a syntactic preference criterion among conflicting arguments, favoring those arguments that are *more informed* or *more direct* [35, 36]. However, other alternative preference criteria could also be used [18]. Given an argument $\langle \mathcal{A}, Q\rangle$, the definitions of counterargument and defeat allows to detect whether other possible arguments $\langle \mathcal{B}_1, Q_1\rangle$,…,$\langle \mathcal{B}_k, Q_k\rangle$ are defeaters for $\langle \mathcal{A}, Q\rangle$. Should the argument $\langle \mathcal{A}, Q\rangle$ be defeated, then it would be no longer supporting its conclusion Q. However, since defeaters are arguments, they may on their turn be defeated. That prompts for a complete recursive dialectical analysis to determine which arguments are ultimately defeated. To characterize this process we will introduce some auxiliary notions.

An *argumentation line* starting in $\langle \mathcal{A}_0, Q_0\rangle$ (denoted $\lambda^{\langle \mathcal{A}_0, Q_0\rangle}$) is a sequence $[\langle \mathcal{A}_0, Q_0\rangle, \langle \mathcal{A}_1, Q_1\rangle, \langle \mathcal{A}_2, Q_2\rangle, \ldots, \langle \mathcal{A}_n, Q_n\rangle, \ldots]$ that can be thought of as an exchange of arguments between two parties, a *proponent* (evenly-indexed arguments) and an *opponent* (oddly-indexed arguments). Each argument $\langle \mathcal{A}_i, Q_i\rangle$ is a defeater for the previous argument $\langle \mathcal{A}_{i-1}, Q_{i-1}\rangle$ in the sequence, $i > 0$. In order to avoid *fallacious* reasoning, dialectical constraints are imposed on such an argument exchange to be considered rationally acceptable in light of a given program $\mathcal{P}$ (viz. disallowing circular argumentation, enforcing the use of proper defeaters to defeat blocking defeaters, etc.[4])

[3] Specificity [35] is used in DeLP as a syntax-based criterion among conflicting arguments, preferring those arguments which are *more informed* or *more direct* [35, 36]. However, other alternative partial orders could also be used.

[4] For an in-depth treatment of DeLP the reader is referred to [18].

Given a program $\mathcal{P}$ and an initial argument $\langle \mathcal{A}_0, Q_0 \rangle$, the set of all acceptable argumentation lines starting in $\langle \mathcal{A}_0, Q_0 \rangle$ accounts for a whole dialectical analysis for $\langle \mathcal{A}_0, Q_0 \rangle$ (i.e., all possible dialogues rooted in $\langle \mathcal{A}_0, Q_0 \rangle$), formalized as a *dialectical tree* $\mathcal{T}_{\langle \mathcal{A}_0, Q_0 \rangle}$. Nodes in a dialectical tree $\mathcal{T}_{\langle \mathcal{A}_0, Q_0 \rangle}$ can be marked as *undefeated* and *defeated* nodes (U-nodes and D-nodes, resp.): all leaves in $\mathcal{T}_{\langle \mathcal{A}_0, Q_0 \rangle}$ will be marked as U-nodes (as they have no defeaters), and every inner node is to be marked as *D-node* iff it has at least one U-node as a child, and as *U-node* otherwise. An argument $\langle \mathcal{A}_0, Q_0 \rangle$ is ultimately accepted as valid (or *warranted*) with respect to a DeLP program $\mathcal{P}$ iff the root of its associated dialectical tree $\mathcal{T}_{\langle \mathcal{A}_0, Q_0 \rangle}$ is labeled as *U-node*.

Solving a query Q with respect to a given program $\mathcal{P}$ accounts for determining whether Q is supported by a warranted argument. Different doxastic attitudes are distinguished when answering query q according to the associated status of warrant: Yes (accounts for believing Q iff there is at least one warranted argument supporting Q on the basis of $\mathcal{P}$); No (accounts for believing $\sim Q$ iff there is at least one warranted argument supporting $\sim Q$ on the basis of $\mathcal{P}$); $Undecided$ (neither Q nor $\sim Q$ are warranted wrt $\mathcal{P}$); and $Unknown$ (Q does not belong to the signature of $\mathcal{P}$).

5 Agent-mediated IK Distribution through the integration of JITIK and DeLP

The JITIK framework presented in Section 2 provides a suitable agent-based platform for knowledge dissemination, taking into consideration hierarchies for users and content classification for determining how distribution rules are to be applied. However, in the case of policies with exceptions or competing policies, specialized criteria have to be explicitly encoded in both Site and Personal Agents in the JITIK platform. In many respects such an approach is undesirable. On the one hand, such changes involve modifying the underlying decision algorithm. The correctness of such changes may be difficult to test, as unexpected side-effects might arise for new future cases. On the other hand, the knowledge engineer should be able to encode knowledge as declaratively as possible, including the possibility of representing competing policies. Such knowledge should be independent of the rational procedure for determining which is the winning policy when conflicting situations arise.

In summary, a number of common features can be identified in large organizations which complicate knowledge dissemination, namely:

- Many different *hierarchies* of organization-related concepts: fields, roles, member interests, etc. These hierarchies can have *exceptions*, as sometimes they may come in conflict with organization policies.
- *Members* of the organization assigned to different areas or divisions and one or more roles within the organization structure (e.g. CEO, manager, supervisor), usually within a *personnel hierarchy.* Such hierarchies do not have exceptions, but they assign different decision power or "permissions" to their members, affecting the ultimate outcome of many decision making processes. Each member will also have his/her own *personal preferences* about IK delivery.
- There are different organization policies which prescribe how to proceed when IK items are to be distributed among different members or users. Many of such policies are *defeasible*, specially in the presence of potentially incomplete information concerning metadata and user profiles. As a result competing policies usually emerge, including exceptions at different levels within the organization structure.

To model the above situations in a computationally attractive way our proposal consists of integrating the JITIK framework with DeLP, incorporating distribution policies for Site Agents explicitly in terms of defeasible logic programs. As explained in Section 2, a Site Agent Ag_S is responsible for distributing IK among different Personal Agents Ag_1, $\ldots Ag_n$. We will use DeLP programs $\mathcal{P}_1, \ldots, \mathcal{P}_n$ to represent user preferences associated with these agents, possibly based on trust relationships wrt other agents or parts of the organization. Knowledge in the Site Agent Ag_S will be represented by another program $\mathcal{P}_S$. In contrast with the knowledge available to Personal Agents, $\mathcal{P}_S$ will contain organizational *corporate rules* defining power and trust relationships (hierarchies, declarative power, etc.) as well as (possibly conflicting) policies for IK distribution among personal agents.

Given a list $I = [Item_1, \ldots, Item_i]$ of IK items to be distributed by the Site Agent Ag_S among different Personal Agents $Ag_1, \ldots, Ag_n$, a distinguished predicate $deliver(I, U)$ will be used to determine which items in I are intended to be delivered to a specific user $u \in U$. This will be solved on the basis of a program $\mathcal{P}$ taking into account the Site Agent's knowledge, the metadata corresponding to the incoming items to be distributed and the personal preferences of the Personal Agents involved. This is made explicit in the algorithm shown in Fig. 2. Note that every time a new item i is delivered to a Personal Agent Ag_i, the source $s \in S$ where this item i comes from (probably another Personal Agent) is identified. Every Personal Agent has a built-in *reputation* function to assess the reliability of every possible source s. The reputation of s wrt Ag_i will be increased (resp. decreased) if the items delivered by s to Ag_i are satisfactory (resp. non-satisfactory) according to some acceptance criterion. Should the reputation of s be lower than a given threshold for Ag_i, then s is no longer considered to be a reliable source. [5] Solving queries based on the *deliver* predicate wrt the DeLP inference engine will automate the decision making process for Site Agents, providing a rationally justified decision even for very complex cases, as we will see in Section 6. The complexity of the algorithm in Fig. 2 is clearly polynomial, but of course there is a hidden cost in solving the query $deliver(item, Ag_i)$, which could depend on the number of items and agents involved.

Modelling Power and Trust for Knowledge Distribution

Our previous work on integrating defeasible argumentation with IK distribution was restricted to aspects like corporate hierarchies, domain classifications and individual preferences [11], leaving out a very relevant aspect in modelling large organizations, namely the presence of different levels of *trust* and *empowerment*. In our current proposal we will model these aspects by considering some distinguished sets: a) a set U of *users* (user identifiers); b) a set I (implemented actually as a list) of specific *information items* to be delivered; c) a set S of *information sources* (usually other agents in the organization); d) a set P of *permission levels* (like "everybody", "ceo", etc.); and e) a set F of *fields* or *areas*. Every information item $i \in I$ will have attributes, like a $f \in F$ (which is related to i by the $isAbout(i, f)$ relation) and the *source* of that information item (related to i by the $source(i, s)$ relation, with $s \in S$). A subset $M \subseteq I$ corresponds to *mandatory* items; non-mandatory items are said to be *optional*. We assume that fields in the organization are organized in hierarchies by means of the $subField(f_1, f_2)$ relation, with $f_1, f_2 \in F$. In particular, the *isAbout* relation is based on computing the transitive closure of $subField$.

Users have attributes too, like $permissions(u, p)$ with $u \in U, p \in P$. The organizational hierarchy is established through the $subordinate(l_1, l_2)$ relation, for permission levels l_i, and its transitive closure $depends(l_1, l_2)$. In order to be able to delegate power from an user u_1 to another user u_2 it is required that the user u_2 *depends on* user u_1 according to his/her position in the organization hierarchy. This is captured by the *can_delegate* relation. Trust is modelled using the $relies(u, s, f)$ relation, with $u \in U$,

[5] We provide the reputation function $reputation_{Ag_i}(S)$ just as a conceptual element in our framework. An in-depth treatment of this topic is outside the scope of this paper; the interested reader is referred to the literature in the area (e.g. [34, 33]).

```
ALGORITHM DistributeItems
{Executed by Site Agent $Ag_S$ to decide distribution of items in $I$
according to power & trust information available}
INPUT:  List $I = [item_1, \ldots, item_k]$ of incoming items,
        DeLP programs $\mathcal{P}_S, \mathcal{P}_1, \ldots, \mathcal{P}_n$
OUTPUT: Item distribution to Personal Agents
BEGIN
  $\mathcal{P}'_S := \mathcal{P}_S \cup \{info(item_1), \ldots, info(item_k)\}$
   {Encode incoming items as new facts for Site Agent}
  FOR every item $item \in I$
    FOR every Personal Agent $Ag_i$ supervised by $Ag_S$
      Let $\mathcal{P} = \mathcal{P}'_S \cup \mathcal{P}_i$
      Let $s$ = source of $item$
     IF $reputation_{Ag_i}(S) > Threshold$ THEN
       Using program $\mathcal{P}$, solve query $deliver(item, Ag_i)$
       IF $deliver(Item, Ag_i)$ is warranted
          THEN
            Send message $item$ to agent $Ag_i$
            $reputation_{Ag_i}(S) \leftarrow reputation_{Ag_i}(S) + EvalMsg(item, Ag_i)$
END
```

Figure 2: Algorithm for Knowledge Distribution using DeLP in a Site Agent

$s \in S$ and $f \in F$, meaning that user u is confident about information items coming from source s when those items are about field f. We consider a low-level *reputation* management mechanism (see algorithm in Fig. 2) for numerically adjusting a reputation level; this is reflected in the knowledge represented at the logical level through the $conf$ predicate (see rules d_5 and d_6 in Fig. 2). The function $EvalMsg(item, Ag_i)$ returns a numerical value which allows to increase/decrease the existing reputation agent Ag_i has with respect to the source of $item$ according to some satisfiability measure agent Ag_i has. Thus it could be the case that $EvalMsg(item, Ag_i) = k > 0$ whenever $item$ satisfies Ag_i expectations, and $EvalMsg(item, Ag_i) = k < 0$ otherwise. Should it be the case that $item$ is not actually delivered to Ag_i, then $EvalMsg(item, Ag_i) = 0$ (see discussion in examples in Section 6).

Finally, at the top level of our model we define the $deliver(i, u)$ relation, which indicates that an item i is to be distributed to user u according to the knowledge available for the SA and the particular user profile associated with the user u. Other details can be found in the DeLP code given in Fig.4.

6 A Worked Example

Next we present an illustrative example of the our approach. We assume a typical corporate environment where members (users) could have different rights within the organization (e.g. CEO, managers, supervisors, etc.), belonging to different organization areas (e.g. production, marketing, etc.).

Let us suppose that there are memos (items) which have to be delivered by a Site Agent to different users in the organization. The Site Agent is required to take hierarchies into account, performing inheritance reasoning to make inferences: the manager can give orders to programmers, but programmers cannot give orders to the manager. Note that there could be *exceptions* to such hierarchies, e.g. if the CEO empowers a programmer to decide about software purchase. In our example, IK items made available from the organization to the Site Agent will correspond to different memos, which will be encoded with a predicate $\mathit{info}(Id, A, L, M, S)$, meaning that the memo with identifier Id is about area A and it can be accessed by users of at least level L. Other attributes associated with the memo are whether it is mandatory ($M = 1$) or optional ($M = 0$), and the source of origin S. Thus, the fact $\mathit{info}(id_3, computers, \mathit{manager}, 0, peter) \leftarrow$ indicates that the memo id_3 is about $computers$, it is intended at least for managers, it is not mandatory, and it has been produced by $peter$.

Fig. 4 shows a sample DeLP code associated with a Site and a particular Personal agent. [6] Strict rules s_1 to s_{10} characterize permissions and extract information from memos. Rule s_1 defines that a user P is *allowed* access to item I if he/she has the required *permissions*, which are given as facts (f_7, f_8 and f_9). Permissions are also propagated using the strict rules s_4, s_5 and s_6, where the binary predicate *depends* establishes the organization hierarchy, stating that the first argument person is (transitively) subordinated to the second one. This predicate is calculated as the transitive closure of a basic predicate *subordinate* (defined by facts f_{10} and f_{11}), which establishes subordinate relationships pairwise. Thus, having e.g. granted permissions as CEO allows the CEO to have access to every memo corresponding to lower level permissions. Rule s_7 indicates when an organization member can delegate power on some other member. Delegation of power is also based on subordination relationships. Rule s_2 and s_3 define the predicate $\mathit{isAbout}(I, A)$ as

[6] Note that we distinguish strict rules, defeasible rules, and facts by using s_i, d_i and f_i as clause identifiers, respectively.

an information hierarchy among subfields. The basic case corresponds to a subfield for which specific information is available (rule s_2), otherwise relationships in this hierarchy (facts f_{12}-f_{17}) are used. Finally, rules s_8, s_9 and s_{10} define auxiliary predicates *source*, *mandatory* and *field* (yes/no) which allow to extract these particular attributes from *info* facts, simplifying the subsequent analysis.

Let us now consider the defeasible rules for our Site Agent. Rules d_1-d_3 define when an item I should be delivered to a specific user U: either because it is interesting for U, or because it is mandatory for U, or because it comes from an authorized source. Rule d_4 defines when something is interesting for a given user. Rule d_5-d_7 define when a user relies on a source (another user) wrt some field F. Rules d_5 and d_6 establish a "naïve" transitive perspective on trust: a user U relies on a source (user) S on a field F if U has confidence on S on F according to the available knowledge, or if U has confidence on some other source S', such that S' relies on S. Note that rule d_7 establishes that unreliability is defined as *"not ultimately provable as reliable"* via default negation. Rules d_8-d_{11} define criteria for authorizing a source for delivering information on a field F: either because the source works on F (d_9), or because the source got explicit power delegation from a superior (d_{11}). Rule d_{10} establishes an exception to d_9 (users who falsified reports are not authorized). Facts f_1-f_3 characterize trust relationships (e.g. joe trusts $mike$ about $computers$, but not about $politics$) stored by the Site Agent.[7] Similarly, facts f_4-f_6 characterize explicit authorizations and delegations. Finally, let us consider the DeLP program associated with a particular Personal Agent (e.g. Joe). A number of facts represent Joe's preferences (interest fields), and a defeasible rule d_1' associated with his preferences indicates that he is not interested in memos from unreliable sources.

We will also assume that Joe has some established reputation values for other people acting as sources of information. We will consider three people ($mike$, $peter$, and $mary$), and we will assume that Joe has the same reputation level for everyone of them (i.e. $reputation_{joe}(mike) = reputation_{joe}(peter) = reputation_{joe}(mary) = 0.8$). Let us also assume that $Threshold = 0.5$

[7] Such trust relationships among Personal Agents could be semi-automatically established on the basis of the reputation function mentioned in Section 5, computed by the Site Agent.

Solving Power and Trust Conflicts using Argumentation

For the sake of example, let us assume that there is a list of three information items $[Memo_1, Memo_2, Memo_3]$ (see Fig. 5) corresponding to memos to be distributed by our Site Agent, which encodes organization policies as a DeLP program $\mathcal{P}_S$ (shown in Fig. 4). By applying the algorithm given in Fig. 2, these items will be encoded temporarily as a set $\mathcal{P}_{items} = \{info(Memo_1)$, $info(Memo_2)$, $info(Memo_3)\}$ (facts f_a-f_c in Fig. 4). For the sake of simplicity, we will assume that there is only one single Personal Agent involved, associated with a specific user *joe*, whose role is *manager*. Joe's Personal Agent mirrors his preferences in terms of a DeLP program $\mathcal{P}_{joe} = \{d'_1, f'_1, f'_2, f'_3\}$, which together with $\mathcal{P}_S$ and $\mathcal{P}_{items}$ will provide the knowledge necessary to decide which IK items should be delivered to this specific user. Following the algorithm in Fig. 2, the Site Agent will have to solve the queries $deliver(id_3, joe)$, $deliver(id_2, joe)$ and $deliver(id_1, joe)$ wrt the DeLP program $\mathcal{P}_S \cup \mathcal{P}_{items} \cup \mathcal{P}_{joe}$. We will show next how each of these queries is solved in different examples that show how DeLP deals with conflicts among organization policies and user preferences.

Example 1 *Consider the query* $deliver(id_3, joe)$. *According to the information provided (Fig. 5), the source of this particular item is mary, and* $reputation_{joe}(mary) = 0.8$. *As mary's reputation is greater than the threshold value the Site Agent has (*0.5*), the algorithm in Fig. 2 establishes that this item could be of interest for joe (as far as the reputation is concerned). Consequently, the query* $deliver(id_3, joe)$ *is solved wrt the program* $\mathcal{P}_S \cup \mathcal{P}_{items} \cup \mathcal{P}_{joe}$.

In this case joe is allowed *to receive this item (rule* s_1*), but it is neither of interest for him (rule* d_1*) nor coming from an authorized person (rule* d_3*). However,* id_3 *is mandatory (fact* f_c*), and hence the Site Agent can compute an argument* $\langle \mathcal{A}_1, deliver(id_3, joe) \rangle$*, with*

$$\mathcal{A}_1 = \{\ deliver(id_3, joe) \prec allowed(id_3, joe),\ mandatory(id_3)\ \}$$

This argument has no defeaters, and hence it is warranted. Thus id_3 *will be delivered to joe. The corresponding dialectical tree has one node (Fig 3(i)).*

Suppose now that joe considers item id_3 *is satisfactory according to his information needs. Thus* $EvalMsg(id_3, joe)$ *will be positive, and mary's reputation will be increased.*

Example 2 *Consider the query* $deliver(id_1, joe)$. *According to the information provided (Fig. 5), the source of this particular item is mike, and* $reputation_{joe}(mike) = 0.8$. *As mike's reputation is greater than the threshold value the Site Agent has*

(0.5), the algorithm in Fig. 2 establishes that this item could be of interest for joe (as far as the reputation is concerned). In this case the DeLP inference engine will find the argument $\langle \mathcal{B}_1, deliver(id_1, joe)\rangle$ with

$$\begin{aligned}\mathcal{B}_1 \;=\; \{&deliver(id_1, joe) \mathrel{-\!\!\prec} allowed(id_1, joe), interest(id_1, joe);\\ &interest(id_1, joe) \mathrel{-\!\!\prec} isAbout(id_1, politics), interestField(politics, joe)\}\end{aligned}$$

However, $\langle \mathcal{B}_1, deliver(id_1, joe)\rangle$ is defeated by $\langle \mathcal{B}_2, \sim interest(id_1, joe)\rangle$, with

$$\begin{aligned}\mathcal{B}_2 \;=\; \{&\sim interest(id_1, joe) \mathrel{-\!\!\prec}\; isAbout(id_1, politics),\; interestField(politics, joe),\\ &\qquad\qquad source(id_1, mike),\; \sim relies(joe, mike, politics);\\ &\sim relies(joe, mike, politics) \mathrel{-\!\!\prec} \mathsf{not}\, relies(joe, mike, politics)^{8}\}\end{aligned}$$

(i.e., according to joe's confidence criteria, joe has no confidence on mike when he talks about politics, so the source is unreliable.) In this case, the dialectical tree $\mathcal{T}_{\langle \mathcal{B}_1,\, distribute(id_1, joe)\rangle}$ has two nodes in a single branch (see Fig. 3(ii)). There are no other arguments to consider. Therefore the answer to the query is NO*, and hence the Site Agent will not deliver id_1 to joe.*

As id_1 was not delivered to joe, he cannot assess to what extent this item was satisfactory according to his information needs. Thus $EvalMsg(id_3, joe)$ is zero, and mike's reputation remains unchanged.

Example 3 *Consider the query $deliver(id_2, joe)$. According to the information provided (Fig. 5), the source of this particular item is peter, and $reputation_{joe}(peter) = 0.8$. Again, as peter's reputation is greater than the threshold value the Site Agent has, the algorithm in Fig. 2 establishes that this item could be of interest for joe. Although joe is* allowed *to receive this item (s_1), note that it is neither of interest for joe (d_1) nor mandatory (d_2). However, there is an argument $\langle \mathcal{C}_1, deliver(id_2, joe)\rangle$ which provides a tentative reason to deliver id_2 to joe, with*

$$\begin{aligned}\mathcal{C}_1 \;=\; \{\qquad deliver(id_2, joe) &\mathrel{-\!\!\prec} allowed(id_2, joe),\\ &\quad authorized\_deliver(id_2, joe);\\ authorized\_deliver(id_2, joe) &\mathrel{-\!\!\prec} source(id_2, peter),\\ &\quad field(id_2, hardware),\\ &\quad isauthorized(peter, hardware)\ ;\\ isauthorized(peter, hardware) &\mathrel{-\!\!\prec} worksOn(peter, hardware)\ \}.\end{aligned}$$

However the argument $\langle \mathcal{C}_1, deliver(id_2, joe)\rangle$ has as a defeater another argument, namely $\langle \mathcal{C}_2, \sim isauthorized(peter, hardware)\rangle$, with

$$\begin{aligned}\mathcal{C}_2 \;=\; \{\sim isauthorized(peter, hardware) &\mathrel{-\!\!\prec} worksOn(peter, hardware),\\ &\qquad falsified\_reports(peter)\ \}.\end{aligned}$$

(peter falsified reports, hence he should not be authorized). However this second argument is superseded by dana's delegation, as there is a third argument $\langle \mathcal{C}_3, isauthorized(peter, hardware)\rangle$, where

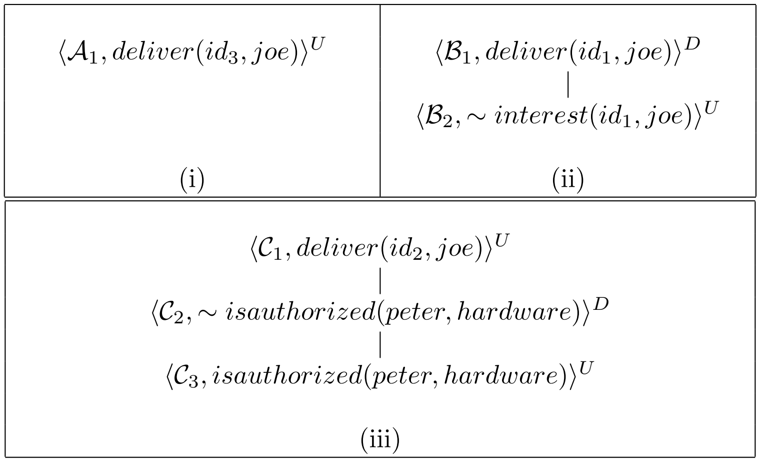
$\langle \mathcal{A}_1, deliver(id_3, joe)\rangle^U$
(i)
$\langle \mathcal{B}_1, deliver(id_1, joe)\rangle^D$
$\langle \mathcal{B}_2, \sim interest(id_1, joe)\rangle^U$
(ii)
$\langle \mathcal{C}_1, deliver(id_2, joe)\rangle^U$
$\langle \mathcal{C}_2, \sim isauthorized(peter, hardware)\rangle^D$
$\langle \mathcal{C}_3, isauthorized(peter, hardware)\rangle^U$
(iii)

Figure 3: Dialectical trees for queries $deliver(id_1, joe)$, $deliver(id_2, joe)$ and $deliver(id_3, joe)$ (examples 1,2 and 3)

$$\begin{array}{lll} \mathcal{C}_3 = & \{isauthorized(peter, hardware) \mathrel{-\!\!\prec} & authorized(dana, hardware), \\ & & delegates(dana, peter), \\ & & permissions(dana, ceo), \\ & & permissions(peter, everybody), \\ & & can\_delegate(ceo, everybody)\ \}. \end{array}$$

In this case, the dialectical tree $\mathcal{T}_{\langle \mathcal{C}_1,\, distribute(id_2, joe)\rangle}$ has three nodes (as shown in Fig. 3(iii)). Therefore the answer to the query is Yes*, and the Site Agent will deliver id_2 to joe. Suppose now that joe considers item id_2 is* not *satisfactory according to his information needs. Thus $EvalMsg(id_2, joe)$ will be negative, and peter's reputation will be decreased. If peter keeps sending items which do not satisfy joe's information need, he might be eventually disregarded as a reliable source (when $reputation_{joe}(peter) \leq Threshold$).*

7 Implementation issues & Ongoing research

Performing defeasible argumentation is a computationally complex task. An abstract machine called JAM (Justification Abstract Machine) has been specially developed for an efficient implementation of DeLP [18]. A full-fledged implementation of DeLP is freely available online[9], including facilities for visualizing arguments and dialectical trees. Several features leading to efficient

[9] See `http://lidia.cs.uns.edu.ar/DeLP`

Site Agent Knowledge

$$\begin{array}{lrcl}
s_1) & allowed(I,U) & \leftarrow & info(I,A,L,M,S),\ permissions(U,L). \\
s_2) & isAbout(I,A) & \leftarrow & info(I,A,L,M,T,S) \\
s_3) & isAbout(I,A) & \leftarrow & subField(SuperA,A),\ isAbout(I,SuperA). \\
s_4) & permissions(U,X) & \leftarrow & depends(X,Y),\ permissions(U,Y). \\
s_5) & depends(X,Y) & \leftarrow & subordinate(X,Y). \\
s_6) & depends(X,Z) & \leftarrow & subordinate(Y,Z),\ depends(X,Y). \\
s_7) & can\_delegate(U1,U2) & \leftarrow & depends(U2,U1). \\
s_8) & source(I,S) & \leftarrow & info(I,\_,\_,\_,S). \\
s_9) & mandatory(I) & \leftarrow & info(I,\_,\_,1,\_). \\
s_{10}) & field(I,F) & \leftarrow & info(I,F,\_,\_,\_). \\
d_1) & deliver(I,U) & \mathrel{-\!\!\prec} & allowed(I,U),\ interest(I,U). \\
d_2) & deliver(I,U) & \mathrel{-\!\!\prec} & allowed(I,U),\ mandatory(I). \\
d_3) & deliver(I,U) & \mathrel{-\!\!\prec} & allowed(I,U),\ authorized\_deliver(I,U). \\
d_4) & interest(I,U) & \mathrel{-\!\!\prec} & isAbout(I,A),\ interestField(A,U). \\
d_5) & relies(U,S,F) & \mathrel{-\!\!\prec} & conf(U,S,F). \\
d_6) & relies(U,S,F) & \mathrel{-\!\!\prec} & conf(U,S1,F), relies(S1,S,F). \\
d_7) & \sim relies(U,S,F) & \mathrel{-\!\!\prec} & \text{not}\ relies(U,S,F). \\
d_8) & authorized\_deliver(I) & \mathrel{-\!\!\prec} & source(I,S), field(I,F), isauthorized(S,F). \\
d_9) & isauthorized(S,F) & \mathrel{-\!\!\prec} & worksOn(S,F). \\
d_{10}) & \sim isauthorized(S,F) & \mathrel{-\!\!\prec} & worksOn(S,F),\ falsified\_reports(S). \\
d_{11}) & isauthorized(S,F) & \mathrel{-\!\!\prec} & authorized(S1,F),\ delegates(S1,S), \\
 & & & permissions(S,P), permissions(S1,P1), \\
 & & & can\_delegate(P1,P).
\end{array}$$

Facts about Confidence, Authorizations, and Power delegation

$$\begin{array}{lr}
f_1) & conf(joe,mike,computers). \\
f_2) & \sim conf(joe,mike,politics) \\
f_3) & conf(mike,bill,computers).
\end{array}$$

$$\begin{array}{lr}
f_4) & \sim authorized(peter,software). \\
f_5) & authorized(dana,hardware). \\
f_6) & delegates(dana,peter). \\
f_x) & falsified\_reports(peter).
\end{array}$$

Facts about Permissions, Roles and Hierarchies

$$\begin{array}{lrc}
f_7) & permissions(joe,manager) & \leftarrow \\
f_8) & permissions(peter,everybody) & \leftarrow \\
f_9) & permissions(dana,ceo) & \leftarrow \\
f_{10}) & subordinate(everybody,manager) & \leftarrow \\
f_{11}) & subordinate(manager,ceo) & \leftarrow \\
f_{12}) & subField(hardware,computers) & \leftarrow \\
f_{13}) & subField(processors,hardware) & \leftarrow \\
f_{14}) & subField(software,computers) & \leftarrow \\
f_{15}) & subField(programing,software) & \leftarrow \\
f_{16}) & subField(computers,infotopics) & \leftarrow \\
f_{17}) & subField(politics,infotopics) & \leftarrow \\
f_{18}) & worksOn(peter,software) & \leftarrow
\end{array}$$

Figure 4: DeLP code for a Site Agent in JITIK

Information Items as facts

$$\boxed{\begin{array}{lrl} f_a) & info(id_1, politics, everybody, 0, mike) & \leftarrow \\ f_b) & info(id_2, hardware, manager, 0, peter) & \leftarrow \\ f_c) & info(id_3, processors, manager, 1, mary) & \leftarrow \end{array}}$$

Personal Agent Knowledge – user preferences

$$\boxed{\begin{array}{lrl} d'_1) & \multicolumn{2}{l}{\sim interest(I, joe) \mathrel{-\!\!\prec} isAbout(I, A),\ interestField(A, joe),\ source(I, S),} \\ & \multicolumn{2}{r}{\sim relies(joe, S, A).} \\ f'_1) & interestField(computers, joe) & \leftarrow \\ f'_2) & \sim interestField(hardware, joe) & \leftarrow \\ f'_3) & interestField(politics, joe) & \leftarrow \end{array}}$$

Figure 5: DeLP code for incoming Information Facts and Preferences in a Personal Agent in JITIK

implementations of DeLP have also been recently studied, in particular those related to computing dialectical trees efficiently [16] and extending DeLP to incorporate possibilistic reasoning [15].

One aspect which deserves particular attention is the comparison between DeLP knowledge encoding capabilities and other rule-based systems, which have the well-known disadvantage of being *brittle* [20], as adding or deleting one rule may substantially change the behavior of the system. In this respect, DeLP inherits all declarative features from logic programming, and as such is highly elaboration tolerant: Answers are always supported by arguments built on the basis of a given DeLP program. Clearly, it is possible that a change in the program (such as the addition or deletion of some program rules) might change the answer obtained as an output for a given query, but any of such changes can be perfectly traced back and justified by showing the underlying dialectical analysis that led to this change.

An implementation of the IK distribution system that contains the Site Agent, the Personal Agents, an Ontology Agent and various Service Agents (web monitoring and others) has been reported elsewhere [4], using the Jade agent platform [2]. Our experiments regarding this integration of IK distribution with defeasible argumentation for modelling power and trust relationships only account as a "proof of concept" prototype, as we have not been able yet to carry out thorough evaluations in the context of a real-world application. In particular, the sample problem presented in Section 6 was encoded and solved successfully using a Java-based DeLP environment.[10]

Part of our current work is focused on adapting the approach proposed in this paper into a truly distributed algorithm, which could improve both the multiagent nature of the procedure as well as its performance and scalability.

8 Related work

In this paper we have extended the previous proposal in [11] for knowledge distribution in large organizations by incorporating the representation of power and trust capabilities explicitly by means of defeasible logic programming. As we have shown, the main advantage obtained is an increased flexibility in modelling different normative situations in organizations and trust relationships. Potentially contradictory knowledge involved in such aspects is suitably handled by the DeLP inference engine.

To the best of our knowledge there are no similar approaches which combine argumentation and agent-mediated knowledge management as the one presented in this paper. A somehow related research is reported in [28] about methods for helping in decision-making processes using argumentation. Besides the differences in the intended application of this system there is also a significative difference in the approach as they use static predefined argumentation schemas, whereas here we propose a general method for constructing arguments that is not restricted to a finite number of argument structures. An interesting direction relating organizational decision making and argumentation is explored in an integrated framework in [25, 24]. This framework, in contrast with our approach, allows for distributed and asynchronous collaboration and aims at giving an active role to the decision makers involved in the solution of the underlying problems. Argumentation is used as a tool within a collaborative decision making process. A discussion moderator (acting as a trusted-third-party) intervenes and, according to a rule-based meta-model of organizational problems, selects the appropriate models whenever a new problem is brought for discussion. Other works related to ours involve decision making and negotiation using argumentation among agents [31]. In contrast, in our system the argumentation process itself is not distributed, taking always place in a central DeLP inference engine.

As we pointed out in Section 5, our approach to modelling a reputation function and providing numerical values representing trust is rather con-

[10] See `http://cs.uns.edu.ar/∼ags/DLP/` for details.

ceptual, as the focus of our analysis is the use of argumentative reasoning for establishing whether a *deliver* literal is warranted or not (based on the procedure described in Fig. 2). Formalizing reputation and trust in multiagent systems has been analyzed in a more broad perspective in the literature. In [9], for example, a functional ontology of reputation for agents is proposed. Such ontology allows to put together the broad knowledge about reputation produced in some areas of interest (mainly in multi-agent systems) and to represent that knowledge in a structured form. In a different perspective, Castelfranchi *et. al* [17, 10] have analyzed different aspects of trust, notably *trust dynamics*. In the latter the authors have considered how direct experiences involving trust, with their successes or failures, influence the future trust of an agent about similar facts. In fact, they challenge the "obvious" approach offered in our formalization (that success always increases trust while failure decreases it), claiming instead that a cognitive attribution process is needed in order to update trust on the basis of an interpretation of the outcome of A's reliance on B and of B's performance (failure or success). They also analyze what happens with the trustworthiness of an agent B in a situation X based on the fact that there is a trust relationship between another agent A with B such that A relies on B in that situation X. A formal model for such trust dynamics is provided, which goes much farther than our conceptual proposal. However, in their approach the authors do not delve into the analysis of knowledge distribution as done here, nor in the use of arguments for supporting rationally justified beliefs.

9 Conclusions

We have presented a novel argument-based approach for supporting IK-distribution processes in large organizations. Our proposal is based on integrating the JITIK framework for Agent-mediated Knowledge Management and Defeasible Logic Programming, a multi-purpose programming language with powerful representation features which provides argumentative reasoning capabilities. We have shown how power relations as well as delegation and trust, can be embedded within our framework, in such a way that a dialectical argumentation process works as a decision core. Conflicts among policies are solved on the basis of a dialectical analysis whose outcome determines to which specific users different pieces of knowledge are to be delivered.

The main advantage obtained by the use of an argumentation engine is an increased flexibility, as it is not necessary to explicitly encode actions for every possible situation. This is particularly important in corporate environments with potentially conflicting information distribution criteria. Our approach is applicable in general to the distribution of IK that can be characterized by symbolic metadata expressed as ground terms in predicate logic.

In the near future we also intend to extend our prototype to handle different real-world institutions with complex normative structures, including also reasoning capabilities for combining argumentative inference with possibilistic reasoning and vague knowledge, as described in [15]. Such formalization allows to have weights associated with defeasible rules, so that defeasible rules with greater weights values would offer a stronger support than those which have smaller values. In such setting an interesting possibility to explore is to introduce reinforcement learning techniques [29] which allow to adapt the weights associated with defeasible rules according to the frequency with which they have led to warranted conclusions, or the reputation or trust associated with some literal in the rule (e.g. in those rules related to sources of incoming items).[11] Research in this direction is currently being pursued.

Acknowledgments The authors would like to thank anonymous reviewers for their suggestions. This work was supported by the Monterrey Tech CAT-011 research chair, by Projects TIC2003-00950, TIN 2004-07933-C03-03, by Ramón y Cajal Program (MCyT, Spain) and by CONICET (Argentina).

References

[1] J.L. Aguirre, R. Brena, and F.J. Cantu. Multiagent-based knowledge networks. *Expert Systems with Applications*, 20(1):65–75, Jan 2001.

[2] F. Bellifemine, A. Poggi, and G. Rimassa. Developing multi-agent systems with jade. In C. Castelfranchi and Y. Lesperance, editors, *Intelligent Agents VII. Agent Theories, Architectures and Languages, 7th International Workshop, ATAL-2000 Proceedings*, LNAI. Springer Verlag, 2001.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[4] R. Brena, J. L. Aguirre, and A. C. Trevino. Just-in-time information and knowledge: Agent technology for km bussiness process. In *Proc. of the 2001 IEEE Conf. on Systems, Man and Cybernetics*. IEEE Press, 2001.

[11] Tom Mitchell – Personal communication.

[5] R. Brena, J. L. Aguirre, and A. C. Trevino. Just-in-time knowledge flow for distributed organizations using agents technology. In *Proc. of the 2001 Knowledge Technologies 2001 Conf., Austin, Texas, 4-7 March 2001*, 2001.

[6] R. Brena and H. Ceballos. A hybrid local-global approach for handling ontologies in a multiagent system. In V. Sgurev R. Yager, editor, *Proc. of the 2004 Intelligent Systems International Conference, Varna, Bulgaria*, pages 261–266. IEEE Press, 2004.

[7] D. Carbogim, D. Robertson, and J. Lee. Argument-based applications to knowledge engineering. *The Knowledge Engineering Review*, 15(2):119–149, 2000.

[8] J. Carrillo. Managing knowledge-based value systems. *Journal of Knowledge Management*, 1(4), June 1998.

[9] S. Casare and J. Simão Sichman. Towards a functional ontology of reputation. In *Proc. of the 4th AAMAS*, pages 505–511, 2005.

[10] C. Castelfranchi, R. Falcone, and G. Pezzulo. Trust in information sources as a source for trust: a fuzzy approach. In *Proc. of the 2nd. AAMAS*, pages 89–96, 2003.

[11] C. Chesñevar, R. Brena, and J. Aguirre. Knowledge distribution in large organizations using defeasible logic programming. In *Proc. 18th Canadian Conf. on AI (in LNCS 3501, Springer)*, pages 244–256, 2005.

[12] C. Chesñevar, R. Brena, and J. Aguirre. Modelling power and trust for knowledge distribution: an argumentative approach. In A. Gelbukh et. al, editor, *LNAI Series (Proc. 3rd MICAI Conf.)*, volume 3789, pages 98–108. Springer, November 2005.

[13] C. Chesñevar, A. Maguitman, and R. Loui. Logical Models of Argument. *ACM Computing Surveys*, 32(4):337–383, December 2000.

[14] C. Chesñevar, A. Maguitman, and G. Simari. Argument-based critics and recommenders: A qualitative perspective on user support systems. *Data & Knowledge Engineering (in press)*, 2006.

[15] C. Chesñevar, G. Simari, T. Alsinet, and L. Godo. A Logic Programming Framework for Possibilistic Argumentation with Vague Knowledge. In *Proc. Intl. Conf. in Uncertainty in Artificial Intelligence (UAI), Canada*, pages 76–84, July 2004.

[16] C. Chesñevar, G. Simari, and L. Godo. Computing dialectical trees efficiently in possibilistic defeasible logic programming. *LNAI Springer Series Vol. 3662 (Proc. of the 8th LPNMR Conf.)*, pages 158–171, September 2005.

[17] R. Falcone and C. Castelfranchi. Trust dynamics: How trust is influenced by direct experiences and by trust itself. In *Proc. of the 3rd. AAMAS*, pages 740–747. IEEE Computer Society, 2004.

[18] A. García and G. Simari. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.

[19] J. Gelati, G. Governatori, A. Rotolo, and G. Sartor. Declarative power, representation and mandate: A formal analysis. In *Proceedings of 15th Conf. on Legal Knowledge and Inf. Systems*, pages 41–52. IOS Press, 2002.

[20] J. Giarratano and G. Riley. *Expert Systems: Principles and Programming.* PWS Publishing Company, 3rd. edition, 1998.

[21] S. Gómez and C. Chesñevar. A Hybrid Approach to Pattern Classification Using Neural Networks and Defeasible Argumentation. In *Proc. of 17th Intl. FLAIRS Conf. Miami, Florida, USA*, pages 393–398. AAAI Press, May 2004.

[22] F. Horibe. *Managing Knowledge Workers.* John Wiley and Sons, 1999.

[23] I. Horrocks. DAML+OIL: a description logic for the semantic web. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, 25(1):4–9, March 2002.

[24] N. Karacapilidis and E. Adamides. Integrating simulation and argumentation in organizational decision making. In *Proc. of the 7th KES Intl. Conf. (in LNAI 2774, Springer), year = 2003, pages = 107-114.*

[25] N. Karacapilidis, E. Adamides, and C. Evangelou. Leveraging organizational knowledge to formulate manufacturing strategy. In *Proc. of the 11th ECIS Conf.*, 2003.

[26] J. Liebowitz and T. Beckman. *Knowledge Organizations.* St. Lucie Press, 1998.

[27] J. Liebowitz and L. Wilcox. *Knowledge Management.* CRC Press, 1997.

[28] J. Lowrance, Harrison W. Ian, and W. Rodriguez. Structured argumentation for analysis. In *Procs. of the 12th Intl. Conf. on Systems Research, Informatics, and Cybernetics*, pages 47–57, Baden-Baden, Germany, Aug 2000.

[29] Tom Mitchell. *Machine Learning.* Mc Graw Hill, 1997.

[30] L. Mui. *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks.* PhD thesis, MIT, 2003.

[31] S. Parsons, C. Sierrra, and N. Jennings. Agents that Reason and Negotiate by Arguing. *Journal of Logic and Computation*, 8:261–292, 1998.

[32] H. Prakken and G. Vreeswijk. Logical Systems for Defeasible Argumentation. In D. Gabbay and F.Guenther, editors, *Handbook of Phil. Logic*, pages 219–318. Kluwer, 2002.

[33] S. Ramchurn, N. Jennings, C. Sierra, and L. Godo. Devising a trust model for multiagent interactions using confidence and reputation. *Applied Artificial Intelligence*, 18(9-10):833–852, 2004.

[34] J. Sabater and C. Sierra. REGRET: reputation in gregarious societies. In Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 194–195, Montreal, Canada, 2001. ACM Press.

[35] G. Simari and R. Loui. A Mathematical Treatment of Defeasible Reasoning and its Implementation. *Art. Intelligence*, 53:125–157, 1992.

[36] F. Stolzenburg, A. García, C. Chesñevar, and G. Simari. Computing Generalized Specificity. *J. of Non-Classical Logics*, 13(1):87–113, 2003.