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

Carlos I. Chesñevar\textsuperscript{1}, Ramón Brena\textsuperscript{2} and José L. Aguirre\textsuperscript{2}  \\ \textsuperscript{1}Artificial Intelligence Research Group – Universitat de Lleida  \\ 25001 Lleida, SPAIN – cic@eps.udl.es  \\ \textsuperscript{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 Program-
ming (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
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 $p : I \rightarrow \wp(U)$. 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 $(i, \{\ldots, u, \ldots\}) \notin p_2$, or vice versa. 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 $p' \in P$.

\(^1\)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 multi-purpose 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 \(\Pi\) and \(\Delta\) stand for sets of strict and defeasible knowledge, respectively. The set \(\Pi\) 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 \(\Delta\) 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 \(\text{not}\) 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 \(P\), an argument \(A\) for a query \(q\), denoted \(\langle A, q \rangle\), is a subset of ground instances of defeasible rules in \(P\) and a (possibly empty) set of default ground literals “\(\text{not} L\)”, such that: 1) there exists a defeasible derivation for \(q\) from \(\Pi \cup A\); 2) \(\Pi \cup A\) is non-contradictory (i.e, \(\Pi \cup A\) does not entail two complementary literals \(p\) and \(\sim p\) (or \(p\) and \(\text{not} p\))), and 3) \(A\) is minimal with respect to set inclusion.

An argument \(\langle A_1, Q_1 \rangle\) is a sub-argument of another argument \(\langle A_2, Q_2 \rangle\) if \(A_1 \subseteq A_2\). Given a DeLP program \(P\), \(\text{Args}(P)\) denotes the set of all possible arguments that can be derived from \(P\).

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 \(\text{no}_p\). 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 \(A\) whenever \(\Pi \cup A\) 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 \( \langle A_1, q_1 \rangle \) is a counterargument for an argument \( \langle A_2, q_2 \rangle \) iff (1) There is a subargument \( \langle A, q \rangle \) of \( \langle A_2, q_2 \rangle \) such that the set \( \Pi \cup \{q_1, q\} \) is contradictory; (2) A literal \( \neg q_1 \) is present in some rule in \( A_1 \). A partial order \( \preceq \subseteq \text{Args}(P) \times \text{Args}(P) \) will be used as a preference criterion among conflicting arguments. An argument \( \langle A_1, q_1 \rangle \) is a defeater for an argument \( \langle A_2, q_2 \rangle \) if \( \langle A_1, q_1 \rangle \) counterargues \( \langle A_2, q_2 \rangle \), and \( \langle A_1, q_1 \rangle \) is preferred over \( \langle A_2, q_2 \rangle \) wrt \( \preceq \). For cases (1) and (2) above, we distinguish between proper and blocking defeaters as follows:

- In case (1) the argument \( \langle A_1, q_1 \rangle \) will be called a proper defeater for \( \langle A_2, q_2 \rangle \) iff \( \langle A_1, q_1 \rangle \) is strictly preferred over \( \langle A, q \rangle \) wrt \( \preceq \).

- In case (1), if \( \langle A_1, q_1 \rangle \) and \( \langle A, q \rangle \) are unrelated to each other, or in case (2), \( \langle A_1, q_1 \rangle \) will be called a blocking defeater for \( \langle A_2, q_2 \rangle \).

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 \( \langle A, Q \rangle \), the definitions of counterargument and defeat allows to detect whether other possible arguments \( \langle B_1, Q_1 \rangle, \ldots, \langle B_k, Q_k \rangle \) are defeaters for \( \langle A, Q \rangle \). Should the argument \( \langle A, Q \rangle \) 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 \( \langle A_0, Q_0 \rangle \) (denoted \( \lambda^{(A_0, Q_0)} \)) is a sequence \( [\langle A_0, Q_0 \rangle, \langle A_1, Q_1 \rangle, \langle A_2, Q_2 \rangle, \ldots, \langle A_n, Q_n \rangle, \ldots] \) 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 \( \langle A_i, Q_i \rangle \) is a defeater for the previous argument \( \langle A_{i-1}, Q_{i-1} \rangle \) 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 \( P \) (viz. disallowing circular argumentation, enforcing the use of proper defeaters to defeat blocking defeaters, etc.)

---

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 $\langle A_0, Q_0 \rangle$, the set of all acceptable argumentation lines starting in $\langle A_0, Q_0 \rangle$ accounts for a whole dialectical analysis for $\langle A_0, Q_0 \rangle$ (i.e., all possible dialogues rooted in $\langle A_0, Q_0 \rangle$), formalized as a dialectical tree $T_{\langle A_0, Q_0 \rangle}$. Nodes in a dialectical tree $T_{\langle A_0, Q_0 \rangle}$ can be marked as undefeated and defeated nodes (U-nodes and D-nodes, resp.): all leaves in $T_{\langle A_0, Q_0 \rangle}$ 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 $\langle A_0, Q_0 \rangle$ is ultimately accepted as valid (or warranted) with respect to a DeLP program $\mathcal{P}$ iff the root of its associated dialectical tree $T_{\langle A_0, Q_0 \rangle}$ 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 $\neg Q$ iff there is at least one warranted argument supporting $\neg Q$ on the basis of $\mathcal{P}$); Undecided (neither $Q$ nor $\neg Q$ 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 $P_1, \ldots, P_n$ 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 $P_S$. In contrast with the knowledge available to Personal Agents, $P_S$ 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 $P$ 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.  

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 $canDelegate$ relation. Trust is modelled using the $relies(u, s, f)$ relation, with $u \in U$.

---

5We 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 \(A_g_S\) 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 \(P_S, P_1, \ldots, P_n\)

**OUTPUT:** Item distribution to Personal Agents

**BEGIN**

\[P'_S := P_S \cup \{info(item_1), \ldots, info(item_k)\}\]
\{Encode incoming items as new facts for Site Agent\}

**FOR** every item \(item \in I\)

**FOR** every Personal Agent \(A_{g_i}\) supervised by \(A_g_S\)

Let \(P = P'_S \cup P_i\)

Let \(s = \text{source of } item\)

**IF** \(\text{reputation}_{A_{g_i}}(S) > \text{Threshold}\) **THEN**

Using program \(P\), solve query \(\text{deliver}(item, A_{g_i})\)

**IF** \(\text{deliver}(item, A_{g_i})\) is warranted

**THEN**

Send message \(item\) to agent \(A_{g_i}\)

\(\text{reputation}_{A_{g_i}}(S) \leftarrow \text{reputation}_{A_{g_i}}(S) + \text{EvalMsg}(item, A_{g_i})\)

**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 \(\text{conf}\) predicate (see rules \(d_5\) and \(d_6\) in Fig. 2). The function \(\text{EvalMsg}(item, A_{g_i})\) returns a numerical value which allows to increase/decrease the existing reputation agent \(A_{g_i}\) has with respect to the source of \(item\) according to some satisfiability measure agent \(A_{g_i}\) has. Thus it could be the case that \(\text{EvalMsg}(item, A_{g_i}) = k > 0\) whenever \(item\) satisfies \(A_{g_i}\) expectations, and \(\text{EvalMsg}(item, A_{g_i}) = k < 0\) otherwise. Should it be the case that \(item\) is not actually delivered to \(A_{g_i}\), then \(\text{EvalMsg}(item, A_{g_i}) = 0\) (see discussion in examples in Section 6).

Finally, at the top level of our model we define the \(\text{deliver}(i, u)\) 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 \( \text{info}(Id, A, L, M, S) \), 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 \( \text{info}(id_3, \text{computers}, \text{manager}, 0, \text{peter}) \) ← indicates that the memo \( id_3 \) is about \text{computers}, it is intended at least for managers, it is not mandatory, and it has been produced by \text{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 \textit{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 \textit{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 \textit{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 \textit{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. 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 $Threshold = 0.5$

\footnote{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 \(P_S\) (shown in Fig. 4). By applying the algorithm given in Fig. 2, these items will be encoded temporarily as a set \(P_\text{items} = \{info(Memo_1), info(Memo_2), info(Memo_3)\}\) (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 \(\text{manager}\). Joe’s Personal Agent mirrors his preferences in terms of a DeLP program \(P_{joe} = \{d'_1, f'_1, f'_2, f'_3\}\), which together with \(P_S\) and \(P_\text{items}\) 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 \(\text{deliver}(id_3, joe)\), \(\text{deliver}(id_2, joe)\) and \(\text{deliver}(id_1, joe)\) wrt the DeLP program \(P_S \cup P_\text{items} \cup P_{joe}\). 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 \(\text{deliver}(id_3, joe)\). According to the information provided (Fig. 5), the source of this particular item is \(mary\), and \(\text{reputation}_{joe}(mary) = 0.8\). 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 \(\text{deliver}(id_3, joe)\) is solved wrt the program \(P_S \cup P_\text{items} \cup P_{joe}\).

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 \(\langle A_1, \text{deliver}(id_3, joe) \rangle\), with

\[ A_1 = \{ \text{deliver}(id_3, joe) \rightarrow \text{allowed}(id_3, joe), \text{mandatory}(id_3) \} \]

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 \(\text{EvalMsg}(id_3, joe)\) will be positive, and \(mary\)’s reputation will be increased.

Example 2 Consider the query \(\text{deliver}(id_1, joe)\). According to the information provided (Fig. 5), the source of this particular item is \(mike\), and \(\text{reputation}_{joe}(mike) = 0.8\). 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 \(\langle B_1, \text{deliver}(id_1, joe) \rangle\) with

\[ B_1 = \{ \text{deliver}(id_1, joe) \leftarrow \text{allowed}(id_1, joe), \text{interest}(id_1, joe); \\
\text{interest}(id_1, joe) \leftarrow \text{isAbout}(id_1, \text{politics}), \text{interestField}(\text{politics}, joe) \} \]

However, \(\langle B_1, \text{deliver}(id_1, joe) \rangle\) is defeated by \(\langle B_2, \sim \text{interest}(id_1, joe) \rangle\), with

\[ B_2 = \{ \sim \text{interest}(id_1, joe) \leftarrow \text{isAbout}(id_1, \text{politics}), \text{interestField}(\text{politics}, joe), \\
\text{source}(id_1, mike), \sim \text{relies}(joe, mike, \text{politics}); \\
\sim \text{relies}(joe, mike, \text{politics}) \leftarrow \text{not\ relies}(joe, mike, \text{politics}) \} \]

(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 \(T_{\langle B_1, \text{distribute}(id_1, joe) \rangle}\) 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 \(\text{deliver}(id_2, joe)\). According to the information provided (Fig. 5), the source of this particular item is peter, and \(\text{reputation}_{joe}(peter) = 0.8\). 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 \(\langle C_1, \text{deliver}(id_2, joe) \rangle\) which provides a tentative reason to deliver id_2 to joe, with

\[ C_1 = \{ \text{deliver}(id_2, joe) \leftarrow \text{allowed}(id_2, joe), \\
\text{authorized}_\text{deliver}(id_2, joe); \\
\text{source}(id_2, peter), \text{field}(id_2, \text{hardware}), \\
\text{isauthorized}(peter, \text{hardware}) \leftarrow \text{worksOn}(peter, \text{hardware}) \}. \]

However the argument \(\langle C_1, \text{deliver}(id_2, joe) \rangle\) has as a defeater another argument, namely \(\langle C_2, \sim \text{isauthorized}(peter, \text{hardware}) \rangle\), with

\[ C_2 = \{ \sim \text{isauthorized}(peter, \text{hardware}) \leftarrow \text{worksOn}(peter, \text{hardware}), \\
\text{falsified}_\text{reports}(peter) \}. \]

(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 \(\langle C_3, \text{isauthorized}(peter, \text{hardware}) \rangle\), where
\[ \langle A_1, deliver(id_3, joe) \rangle^U \]
\[ \langle B_1, deliver(id_1, joe) \rangle^D \]
\[ \langle B_2, \sim interest(id_1, joe) \rangle^U \]

(i)

\[ \langle C_1, deliver(id_2, joe) \rangle^U \]
\[ \langle C_2, \sim isauthorized(peter, hardware) \rangle^D \]
\[ \langle C_3, isauthorized(peter, hardware) \rangle^U \]

(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)

\[ C_3 = \{ isauthorized(peter, hardware) \sim authorized(dana, hardware), \]
\[ delegates(dana, peter), \]
\[ permissions(dana, ceo), \]
\[ permissions(peter, everybody), \]
\[ can_delegate(ceo, everybody) \} \]

In this case, the dialectical tree \( T_{\langle C_1, distribute(id_2, joe) \rangle} \) 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{align*}
s_1) \quad \text{allowed}(I, U) & \leftarrow \text{info}(I, A, L, M, S), \text{permissions}(U, L). \\
s_2) \quad \text{isAbout}(I, A) & \leftarrow \text{info}(I, A, L, M, T, S) \\
s_3) \quad \text{isAbout}(I, A) & \leftarrow \text{subField}(\text{SuperA}, A), \text{isAbout}(I, \text{SuperA}). \\
s_4) \quad \text{permissions}(U, X) & \leftarrow \text{depends}(X, Y), \text{permissions}(U, Y). \\
s_5) \quad \text{depends}(X, Y) & \leftarrow \text{subordinate}(X, Y). \\
s_6) \quad \text{depends}(X, Z) & \leftarrow \text{subordinate}(Y, Z), \text{depends}(X, Y). \\
s_7) \quad \text{canDelegate}(U1, U2) & \leftarrow \text{depends}(U2, U1). \\
s_8) \quad \text{source}(I, S) & \leftarrow \text{info}(I, A, \ldots, S). \\
s_9) \quad \text{mandatory}(I) & \leftarrow \text{info}(I, A, \ldots, 1, \ldots). \\
s_{10}) \quad \text{field}(I, F) & \leftarrow \text{info}(I, F, A, \ldots).
\end{align*}

Facts about Confidence, Authorizations, and Power delegation

\begin{align*}
d_1) \quad \text{deliver}(I, U) & \leftarrow \text{allowed}(I, U), \text{interest}(I, U). \\
d_2) \quad \text{deliver}(I, U) & \leftarrow \text{allowed}(I, U), \text{mandatory}(I). \\
d_3) \quad \text{deliver}(I, U) & \leftarrow \text{allowed}(I, U), \text{authorizedDeliver}(I, U). \\
d_4) \quad \text{interest}(I, U) & \leftarrow \text{isAbout}(I, A), \text{interestField}(A, U). \\
d_5) \quad \text{relies}(U, S, F) & \leftarrow \text{conf}(U, S, F). \\
d_6) \quad \text{relies}(U, S, F) & \leftarrow \text{conf}(U, S1, F), \text{relies}(S1, S, F). \\
d_7) \quad \sim \text{relies}(U, S, F) & \leftarrow \text{not \ relies}(U, S, F). \\
d_8) \quad \text{authorizedDeliver}(I) & \leftarrow \text{source}(I, S), \text{field}(I, F), \text{isAuthorized}(S, F). \\
d_9) \quad \text{isAuthorized}(S, F) & \leftarrow \text{worksOn}(S, F). \\
d_{10}) \quad \sim \text{isAuthorized}(S, F) & \leftarrow \text{worksOn}(S, F), \text{falsifiedReports}(S). \\
d_{11}) \quad \text{isAuthorized}(S, F) & \leftarrow \text{authorized}(S1, F), \text{delegates}(S1, S), \\
& \text{permissions}(S, P), \text{permissions}(S1, P1), \text{canDelegate}(P1, P).
\end{align*}

Facts about Permissions, Roles and Hierarchies

\begin{align*}
f_1) \quad \text{conf}(\text{Joe}, \text{Mike}, \text{Computers}). \\
f_2) \quad \sim \text{conf}(\text{Joe}, \text{Mike}, \text{Politics}). \\
f_3) \quad \text{conf}(\text{Mike}, \text{Bill}, \text{Computers}). \\
f_4) \quad \sim \text{authorized}(\text{Peter}, \text{Software}). \\
f_5) \quad \text{authorized}(\text{Peter}, \text{Hardware}). \\
f_6) \quad \text{delegates}(\text{Peter}, \text{Electronic}). \\
f_7) \quad \text{falsifiedReports}(\text{Peter}).
\end{align*}

Figure 4: DeLP code for a Site Agent in JITIK
Information Items as facts

\[
\begin{align*}
&f_a) \quad \text{info}(id_1, \text{politics}, \text{everybody}, 0, \text{mike}) \leftarrow \\
&f_b) \quad \text{info}(id_2, \text{hardware}, \text{manager}, 0, \text{peter}) \leftarrow \\
&f_c) \quad \text{info}(id_3, \text{processors}, \text{manager}, 1, \text{mary}) \leftarrow
\end{align*}
\]

Personal Agent Knowledge – user preferences

\[
\begin{align*}
&d'_1) \sim \text{interest}(I, \text{joe}) \sim \text{isAbout}(I, A), \text{interestField}(A, \text{joe}), \text{source}(I, S), \\
&\quad \sim \text{relies}(\text{joe}, S, A), \\
&f'_1) \quad \text{interestField}(\text{computers}, \text{joe}) \leftarrow \\
&f'_2) \quad \sim \text{interestField}(\text{hardware}, \text{joe}) \leftarrow \\
&f'_3) \quad \text{interestField}(\text{politics}, \text{joe}) \leftarrow
\end{align*}
\]

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.\textsuperscript{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.

\section{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-

\footnote{See \url{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 multi-agent 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). 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


\[11]\text{Tom Mitchell – Personal communication.}\]


