MAKING ARGUMENT SYSTEMS COMPUTATIONALLY ATTRACTIVE
Argument Construction and Maintenance [1]

Alejandro J. García, Carlos I. Chesñevar, and Guillermo R. Simari [2]

Departamento de Matemática, Universidad Nacional del Sur,
Alem 1253, (8000) Bahía Blanca, ARGENTINA.
Phone: (54) (91) 20776 (ext. 317), FAX: (54) (91) 551447
e-mail: `grs@arcriba.edu.ar`

1. INTRODUCTION

Argumentative systems (Pollock,1987; Vreeswijk, 1989; Prakken, 1993) are formalizations of the process of "defeasible reasoning", *i.e.*, reasoning to reach conclusions that could be discarded when new evidence appears. An argument for a conclusion p is a tentative piece of reasoning an agent would accept to explain p. If the agent gets new information, the conclusion p together with the argument that supported p may no longer be valid. In that way nonmonotonicity arises. The analysis of the relationships among arguments naturally captures many features of commonsense reasoning, which could be unclear or difficult to introduce in other frameworks, such as Default Logic (Reiter, 1980), Nonmonotonic Logic (McDermott & Doyle, 1980), Autoepistemic Logic (Moore, 1985) and Circumscription (McCarthy,1980).

A query q is a request to the system for *justifying* q. The justification process involves the construction of an *acceptable argument* for q from the information stored in the system's knowledge base (KB). To decide the acceptability of an argument A, possible *counterarguments* for A are generated. These counterarguments are in turn tested for acceptability. Those which are *accepted* are then compared with A using a *specificity* relationship, which defines a partial ordering among arguments.

Computing justifications requires considerable effort, therefore it is desirable that the system would be able to save work already done. This repository, an Arguments

[1]This work was partially supported by the Secretaría de Ciencia y Técnica, Universidad Nacional del Sur.

[2]Members of the Artificial Intelligence Research Group (Grupo de Investigación en Inteligencia Artificial, GIIA), Universidad Nacional del Sur, ARGENTINA.

Base, would contain all the justifications the agent has computed in the past and remain valid.

An intelligent agent must be able to act in a changing environment, learning new facts about the world. By incorporating a new fact into the knowledge base, old conclusions might become invalid, and new arguments, or counterarguments, could be obtained. The key to the problem is to detect which of the arguments saved in the Arguments Base will be affected by the addition of that new fact.

This paper describes the implementation issues of a defeasible reasoning system, the ARGUS system, following the Simari and Loui's approach (Simari & Loui, 1992). Our approach includes some novel features such as an Arguments Maintenance System (AMS) to improve the performance of the reasoner, an optimized argument construction procedure, a consistency check procedure embedded in the inference engine, and a pruning strategy for defeasible inference trees. In order to facilitate the specification of the algorithms that implement these features, new concepts and definitions are introduced.

2. ARGUMENTS

In this section we will briefly mention the construction of a formal system $\mathbf{I\!L}$. This formalism will provide a language to represent the knowledge of a given agent $\mathcal{A}$ who will perform her defeasible reasoning through the formulation of tentative arguments using that language (see Simari & Loui (1992) for further details). These arguments will be the subject of a screening process that will establish a preference order on them. Finally, when *counterarguments* are found, they will in turn be *compared* with the original argument using the preference partial order.

The language of $\mathbf{I\!L}$ in which $\mathcal{A}$ will represent her beliefs is composed of a *first order language* $\mathcal{L}$, plus a binary meta-linguistic relation defined on the set of non-closed literals of $\mathcal{L}$. The members of the meta-linguistic relation are called *defeasible rules* and they have the form $\alpha \;{>\!\!-}\; \beta$, where α and β must be non-closed well-formed formulas (*wffs*) in $\mathcal{L}$. The relation " $>\!\!-$ " is understood as expressing that "reasons to believe in the antecedent α provide reasons to believe in the consequent β".

We denote with $Sent(\mathcal{L})$ the set of sentences of $\mathcal{L}$. Let $\mathcal{K}$ be a consistent subset of $Sent(\mathcal{L})$ called the *context*. $\mathcal{K}$ represents the beliefs of $\mathcal{A}$, and can be partitioned in two subsets corresponding to necessary (general) $Sent_{\mathcal{N}}(\mathcal{L})$ and contingent (particular) information $Sent_{\mathcal{C}}(\mathcal{L})$. In mapping $\mathcal{A}$'s reality to the set $\mathcal{K}$ we obtain a partition of it in two subsets $\mathcal{K}_G = Sent_{\mathcal{N}}(\mathcal{L}) \cap \mathcal{K}$ and $\mathcal{K}_P = Sent_{\mathcal{C}}(\mathcal{L}) \cap \mathcal{K}$. Clearly, $\mathcal{K} = \mathcal{K}_G \cup \mathcal{K}_P$.

The beliefs of $\mathcal{A}$ are represented in $\mathbf{I\!L}$ by a pair $(\mathcal{K}, \Delta)$, called *Defeasible Logic Structure*, where Δ is a finite set of defeasible rules. $\mathcal{K}$ represents the non-defeasible part of $\mathcal{A}$'s knowledge and Δ represents information that $\mathcal{A}$ is prepared to take at less than face value. $\Delta^{\downarrow}$ denotes the set of all grounded instances of members of Δ.

Given $(\mathcal{K}, \Delta)$, we need to define when a fact can be regarded as *justified.* A *defeasible derivation* is defined as a derivation where some defeasible rules are used as material implications for the application of *modus ponens.* Let Γ be a subset of $\mathcal{K} \cup \Delta^{\downarrow}$. The grounded literal h is a *defeasible consequence* of Γ, abbreviated $\Gamma \mid\!\sim h$, if and only if there exists a finite sequence $B_1, \ldots, B_n$ such that $B_n = h$ and for $1 \quad i < n$, either $B_i \in \Gamma$, or B_i is a direct consequence of the preceding elements in the sequence by virtue of the application of *modus ponens* or *particularization* (grounding) of a universally quantified sentence. Also, we will write $\mathcal{K} \cup A \mid\!\sim h$ distinguishing the set A of defeasible rules used in the derivation from the context $\mathcal{K}$.

In first order logic the above definition is enough to describe the wffs that are theorems, but we need to give a criterion that will allow us to prefer one conclusion to another. We will now introduce the formal notion of *argument*.

DEFINITION **2.1** Given a context $\mathcal{K} = \mathcal{K}_G \cup \mathcal{K}_P$, a set Δ of defeasible rules, and a literal $h \in Sent_{\mathcal{C}}(\mathcal{L})$, we say that a subset A of $\Delta^{\downarrow}$ is an *argument structure* for h in the context $\mathcal{K}$(denoted by $\langle A, h\rangle_{\mathcal{K}}$, or just $\langle A, h\rangle$) if and only if:
1) $\mathcal{K} \cup A \mathrel{|\!\!\sim} h$,
2) $\mathcal{K} \cup A \not\mathrel{|\!\!\sim} \perp$ and
3) $\nexists A' \subset A,\ \mathcal{K} \cup A' \mathrel{|\!\!\sim} h$.
A *subargument* of $\langle A, h\rangle$ is an argument $\langle S, j\rangle$ such that $S \subseteq A$.

EXAMPLE **2.1** Given $(\mathcal{K}, \Delta)$, $\mathcal{K} = \{d \mathrel{-\!\!>} b, d, f, l\}$, $\Delta = \{b \wedge c \mathrel{>\!\!-} h, f \mathrel{>\!\!-} c, l \wedge f \mathrel{>\!\!-} \neg c\}$, we say that $\langle\{f \mathrel{>\!\!-} c, b \wedge c \mathrel{>\!\!-} h\}, h\rangle$ is an argument structure for h.

We will refer to the collections of all possible argument structures as $\mathbf{AStruc}(\Delta^{\downarrow})$, or just **AStruc**. The following definitions will characterize the relations of *specificity, disagreement, counterargumentation*, and *defeat* on **AStruc**.

DEFINITION **2.2** Let $\mathcal{D} = \{a \in Lit(\mathcal{K} \cup \Delta) : \mathcal{K} \cup \Delta^{\downarrow} \mathrel{|\!\!\sim} a\}$, where $Lit(A)$ is the set of literals in the wff A, and $\langle A_1, h_1\rangle$,$\langle A_2, h_2\rangle \in$ **AStruc**. We say that A_1 for h_1 is *strictly more specific than* A_2 for h_2 denoted $\langle A_1, h_1\rangle \succ_{\mathbf{spec}} \langle A_2, h_2\rangle$, if and only if:
i) $\forall S \subseteq \mathcal{D}$ if $\mathcal{K}_G \cup S \cup A_1 \mathrel{|\!\!\sim} h_1$ and $\mathcal{K}_G \cup S \not\mathrel{|\!\!\sim} h_1$, then $\mathcal{K}_G \cup S \cup A_2 \mathrel{|\!\!\sim} h_2$.
ii) $\exists S \subseteq \mathcal{D}$ such that $\mathcal{K}_G \cup S \cup A_2 \mathrel{|\!\!\sim} h_2$ and $\mathcal{K}_G \cup S \not\mathrel{|\!\!\sim} h_2$ and $\mathcal{K}_G \cup S \cup A_1 \not\mathrel{|\!\!\sim} h_1$.

DEFINITION **2.3** Two argument structures $\langle A_1, h_1\rangle$ and $\langle A_2, h_2\rangle$ *disagree*, denoted $\langle A_1, h_1\rangle \bowtie_{\mathcal{K}} \langle A_2, h_2\rangle$, if and only if $\mathcal{K} \cup \{h_1, h_2\} \vdash \perp$.

DEFINITION **2.4** Given two argument structures $\langle A_1, h_1\rangle$ and $\langle A_2, h_2\rangle$, we say that $\langle A_1, h_1\rangle$ *counterargues* $\langle A_2, h_2\rangle$ in the literal h, denoted $\langle A_1, h_1\rangle \overset{h}{\rightarrow} \langle A_2, h_2\rangle$, if and only if there exists a subargument $\langle A, h\rangle$ of $\langle A_2, h_2\rangle$ such that $\langle A_1, h_1\rangle \bowtie_{\mathcal{K}} \langle A, h\rangle$. $\langle A, h\rangle$ will be also called the *disagreement subargument.*

DEFINITION **2.5** Given two argument structures $\langle A_1, h_1\rangle$ and $\langle A_2, h_2\rangle$, we say that $\langle A_1, h_1\rangle$ *defeats* $\langle A_2, h_2\rangle$ at literal h, denoted $\langle A_1, h_1\rangle \gg_{\mathbf{def}} \langle A_2, h_2\rangle$, if and only if there exists a subargument $\langle A, h\rangle$ of $\langle A_2, h_2\rangle$ such that: $\langle A_1, h_1\rangle \overset{h}{\rightarrow} \langle A_2, h_2\rangle$ and $\langle A_1, h_1\rangle \succ_{\mathbf{spec}} \langle A, h\rangle$.

EXAMPLE **2.2** Given $(\mathcal{K}, \Delta)$ as defined in example 2.1, we have the following relations between arguments.

$$\begin{aligned}
&\langle\{l \wedge f \mathrel{>\!\!-} \neg c\}, \neg c\rangle \bowtie_{\mathcal{K}} \langle\{f \mathrel{>\!\!-} c\}, c\rangle \\
&\langle\{l \wedge f \mathrel{>\!\!-} \neg c\}, \neg c\rangle \overset{c}{\rightarrow} \langle\{f \mathrel{>\!\!-} c, b \wedge c \mathrel{>\!\!-} h\}, h\rangle \\
&\langle\{l \wedge f \mathrel{>\!\!-} \neg c\}, \neg c\rangle \succ_{\mathbf{spec}} \langle\{f \mathrel{>\!\!-} c\}, c\rangle \\
&\langle\{l \wedge f \mathrel{>\!\!-} \neg c\}, \neg c\rangle \gg_{\mathbf{def}} \langle\{f \mathrel{>\!\!-} c, b \wedge c \mathrel{>\!\!-} h\}, h\rangle
\end{aligned}$$

DEFINITION **2.6** An argument $\langle A, h\rangle$ is active at various levels as supporting (S-argument) or interfering argument (I-argument):
i. All arguments are (level 0) S-arguments and I-arguments.
ii. $\langle A_1, h_1\rangle$ is a (level $n + 1$) S-argument if and only if there is no level n I-argument $\langle A_2, h_2\rangle$ such that for some h, $\langle A_2, h_2\rangle$ counterargues $\langle A_1, h_1\rangle$ at h.
iii. $\langle A_1, h_1\rangle$ is a (level $n + 1$) I-argument if and only if there is no level n I-argument $\langle A_2, h_2\rangle$ such that $\langle A_2, h_2\rangle$ defeats $\langle A_1, h_1\rangle$.

Finally, we will say that an argument $\langle A, h\rangle$ is a *justification* for h if and only if there exists m such that for all $n \geq m$ $\langle A, h\rangle$ is an S-argument of level n for h. It can be shown that there is an effective procedure to decide whether $\langle A, h\rangle$ *justifies* h (Simari & Loui, 1992).

3. KNOWLEDGE REPRESENTATION

The system maintains a knowledge base $[\mathcal{K}, \Delta]$ and an Arguments Base $\mathbb{B}$(García, Chesñevar & Simari, 1992). $[\mathcal{K}, \Delta]$ is the computational counterpart of $(\mathcal{K}, \Delta)$. $\mathbb{B}$ stores the justifications already built by the system. The elements of $\mathcal{K}$ are of two kinds: *strong rules* (of the form $literal_1 \wedge literal_2 \wedge \ldots \wedge literal_n \Rightarrow literal$) [3] or *particular facts*, corresponding to grounded literals. Δ is a finite set of *defeasible rules* of the form $literal_1 \wedge \ldots \wedge literal_n \succ\!\!- literal$. $\mathbb{B}$ stores arguments already generated by the system, along with information relating them to other members of $\mathbb{B}$.

The implementation of $[\mathcal{K}, \Delta]$ involves also the definition of "$\succ\!\!-$" (defeasible implication), "$\Rightarrow$" (material implication), "$\wedge$" (conjunction) and "$\neg$" [4] (corresponds to classic negation; the system assumes that $\neg\neg l = l$).

Example **3.1** The following example (Poole, 1988) shows how knowledge can be represented using this formalism. [5]

Bats are mammals.	bat(X) $\Rightarrow$ mammal(X)
Bats normally fly.	bat(X) $\succ\!\!-$ flies(X)
Mammals normally don't fly.	mammal(X) $\succ\!\!- \neg$ flies(X)
Dead bats normally don't fly.	bat(X) $\wedge$ dead(X) $\succ\!\!- \neg$ flies(X)
Dracula is a bat.	bat(dracula)
Dracula is dead.	dead(dracula)

Since the knowledge base $\mathcal{K}$ is a subset of $\mathcal{L}$, the inference engine must consider the contraposition for material implication. When the strong rule $a \Rightarrow b$ is introduced as part of the agent's knowledge, we are also meaning $\neg b \Rightarrow \neg a$. In order to capture this feature, every rule in $\mathcal{K}$ is implemented as a list of literals that represents its clausal form. Thus, a strong rule $l_1 \wedge l_2 \wedge \ldots \wedge l_k \Rightarrow c$ is represented as $[\neg\, l_1, \neg\, l_2, \ldots\; \neg\, l_k, c]$. Actually, a clausal form $[l_1, l_2, \ldots\; l_n]$ represents n strong rules with consequents $l_1, l_2, \ldots l_n$ respectively, where the associated antecedents to each l_i are the literals $l_1, \ldots, l_{i-1}, l_{i+1}, \ldots, l_n$ negated. This representation (Loveland, 1978; Poole, 1985b) captures the meaning of contraposition allowing the inference engine to remain independent of the way rules where added to $\mathcal{K}$. It is important to remark that contraposition is not allowed for defeasible rules.

In the search for a supporting argument for a grounded literal q, the system looks first for an existing justification stored in $\mathbb{B}$. Should this search fail, the system will attempt to build an argument for q from $[\mathcal{K}, \Delta]$. The system also allows the addition of new facts (grounded literals) to the knowledge base $\mathcal{K}$. This action activates the *Arguments Maintenance System* (AMS) that scans the Arguments Base $\mathbb{B}$, eliminating every argument incompatible with the new knowledge base. Then, the AMS builds those new arguments the new fact has made possible. After comparing the new arguments with the ones already stored in $\mathbb{B}$, the relation among them will be properly updated.

[3]A literal is an atomic formula or an atomic formula negated. Atomic formulas are formulas of the form $p(t_1, t_2, \ldots, t_n)$, where p is the predicate name and $t_1, t_2, \ldots, t_n$ are constants or variables.

[4]Presented as *neg* in (Simari & Loui, 1992)

[5]Predicates, variables and constants names follow the syntactic PROLOG convention.

For a given query q, the system will answer:

- "unknown", if no argument for q can be built from $[\mathcal{K}, \Delta]$;
- "yes", if there exists a justification for q;
- "no", if every argument for q is defeated;
- "undefined", if for every argument A for q there exists at least one non-defeated counterargument that is not comparable for specificity with A.

4. THE ARGUMENT CONSTRUCTION PROCEDURE

Inference is defined in terms of *inference trees* (Lin & Shoham, 1989). Using this notion it is possible to redefine the concepts of *defeasible consequence* and *argument structure* in a computational oriented manner.

DEFINITION **4.1** Let q be a goal. Then a *Defeasible Inference Tree (DIT)* for q is defined as follows:
i) A particular fact q is a defeasible inference tree for the goal q.
ii) If $T_1, \ldots, T_n$ are defeasible inference trees with roots $l_1, \ldots, l_n$ respectively, and $l_1 \wedge l_2 \ldots \wedge\ l_n \Rightarrow q$ is a rule in $\mathcal{K}$, such that q is not a node in any of the trees $T_1, \ldots, T_n$, then the tree T with root q and $T_1 \ldots T_n$ as immediate subtrees is a defeasible inference tree for q. We say that T is built from $T_1, \ldots, T_n$ using the strong rule $l_1 \ldots l_n \Rightarrow q$.
iii) If $T_1, \ldots, T_n$ are defeasible inference trees with roots $l_1, \ldots, l_n$ respectively, and $l_1 \wedge\ l_2 \ldots \wedge\ l_n \succ\!\!- q$ is a rule in $\Delta^{\downarrow}$, such that q is not a node in any of the trees $T_1, \ldots, T_n$, then the tree T with root q and $T_1 \ldots T_n$ as immediate subtrees is a defeasible inference tree for q. We say that T is built from $T_1, \ldots, T_n$ using the defeasible rule $l_1 \ldots l_n \succ\!\!- q$.

The defeasible consequence meta-meta-relationship "$\mid\!\sim$ " (Simari & Loui, 1992) can be defined in terms of defeasible consequence trees: we will say that $\Gamma \mid\!\sim q$ if there exists a DIT for q built from the rules in Γ. If $\langle A, h\rangle$ is an argument structure for q, the set A contains the defeasible rules of a DIT with root h. Thus, we can introduce the following definition of argument in terms of a DIT.

DEFINITION **4.2** Let T be a DIT for a literal h, and A the set of defeasible rules used in the construction of T. We say that $\langle A, h\rangle$ is an argument structure for h if: (1) $\mathcal{K}\ \cup\ A \not\mid\!\sim \bot$ and (2) $\not\exists A' \subset A,\ \mathcal{K}\ \cup\ A' \mid\!\sim h$.

The definition 4.2 gives a way to obtain an argument without building $\Delta^{\downarrow}$ as definition 2.1 requires. The system builds a DIT for a grounded literal q using backward chaining, trying to unify q with some rule R from $[\mathcal{K}, \Delta]$. If this unification succeeds, then the antecedents of R become new goals to be satisfied. Unification (Lloyd,1987) is extended to consider defeasible rules. Once the DIT for q has been built, the set A of defeasible rules used in it will be an argument for q (see definition 4.2) when verifies: (1) $\mathcal{K} \cup\ A \not\mid\!\sim \bot$ (consistency) and (2) $\not\exists A' \subset A,\ \mathcal{K} \cup A' \mid\!\sim h$ (minimality). Next we will discuss briefly these two conditions and the pruning strategy used during the construction of defeasible inference trees.

Let $[\mathcal{K}, \Delta]$ be the knowledge base of an agent $\mathcal{A}$, and let $\langle A, h\rangle$ be an argument structure. We will say that $\langle A, h\rangle$ is consistent with respect to $\mathcal{K}$, *i.e.*, $\mathcal{K}\ \cup\ A \not\mid\!\sim \bot$, if and only if there is no $P \in (\mathcal{K}\ \cup\ A)^{\vdash}$ [6] such that $\mathcal{K}\ \cup\ A \mid\!\sim\ P$ and $\mathcal{K}\ \cup\ A \mid\!\sim\ \neg P$.

[6] $R^{\vdash}$ represents the classic deductive closure of R.

PROPOSITION **4.1** Let $\mathcal{K}$ be a consistent set and let $\langle A, h\rangle$ be an argument structure for h and let $l_1 \wedge l_2, \ldots l_n \succ\!\!- c$ be a grounded instance of a defeasible rule in A. If $\mathcal{K} \not\vdash \neg c$, then the rule $l_1 \wedge l_2, \ldots l_n \succ\!\!- c$ can be used to extend $\mathcal{K}$ with c in a consistent way, *i.e.*, $\mathcal{K}' = \mathcal{K} \cup \{c\}$ is consistent. □

The consistency of a DIT for h is checked applying a recursive procedure to each subtree, starting from the leaves and ending in the subtree for h, *i.e.*, the DIT itself. The leaves of a DIT are facts belonging to a consistent $\mathcal{K}$. Proposition 4.1 says that if we start with a consistent knowledge base $\mathcal{K}$, then a defeasible rule $l_1, \ldots l_n \succ\!\!- c$ can be used as valid only if the consequent c can be assumed consistently, *i.e.*, $\mathcal{K} \not\vdash \neg c$. When $\mathcal{K} \vdash \neg c$ the rule must be discarded, and the current subtree must be rebuilt. In this way verification is done only once for each rule and reconstruction is done only when necessary.

EXAMPLE **4.1** Let $\mathcal{K} = \{$ $penguin(X)$ $\rightarrow$ $bird(X)$, $penguin(petete)$, $penguin(X)$ $\rightarrow$ $\neg flies(X)$ $\}$ and $\Delta = \{$ $bird(X)$ $\succ\!\!-$ $flies(X)$ $\}$ be a knowledge base. Then $\langle$ $\{$ $bird(petete)$ $\succ\!\!-$ $flies(petete)$ $\}$, $flies(petete)$ $\rangle$ is *not* an argument structure for $flies(petete)$, since $\mathcal{K} \vdash \neg flies(petete)$.

Given a grounded literal q, the minimality condition in definition 4.2 is checked by building all possible sets $A_1, A_2, \ldots, A_n$ of defeasible rules, such that for every A_i, conditions 1 and 2 of the definition 4.2 hold. The system will discard those sets that have the property of being supersets of any other. The remaining sets of defeasible rules will be arguments for q.

The roots of the subtrees built during the construction of a DIT T for q are recorded locally. Since a ground literal l could be the root of many subtrees of T, this pruning strategy speeds up the construction of an argument A for q by building just one subtree.

5. JUSTIFICATIONS

The process of finding an argument for a ground literal h that results in a justification is quite involved. For a given h, the system's answer will be determined by the posibility of obtaining a justification for h as follows: first, the system will try to build an argument structure $\langle A, h\rangle$ for h from $[\mathcal{K}, \Delta]$. If such $\langle A, h\rangle$ exists, all counterarguments and defeaters for $\langle A, h\rangle$ (if any) are generated. Since defeaters and counterarguments are argument structures, they can have also other defeaters, which can have in turn defeaters, and so on. If every counterargument and every defeater for $\langle A, h\rangle$ is defeated, then the argument A for h becomes a *justification* for h. Nevertheless, if any of the defeaters or counterarguments for $\langle A, h\rangle$ has not been defeated, then the system will try to find another argument which justifies h.

We have formalized this situation in terms of *activation levels* for arguments (see definition 2.6). It has been shown (Simari & Loui, 1992) that there exists a *cut* level such that all the surviving arguments at that level will be active as S-arguments and I-arguments at the next level. This fact guarantees the existence of an effective procedure for the computation of justifications since every S-arguments $\langle A, h\rangle$ active in the cut level justifies h. Nevertheless, this procedure is computationally expensive. For that reason, we will analyze the problem from an alternate point of view of defeasible inference trees.

DEFINITION **5.1** Let $\langle A, h\rangle$ be an argument structure. A *defeaters tree* for $\langle A, h\rangle$, denoted T_D, is recursively defined as follows:

i. An argument structure $\langle A, h\rangle$ with no defeaters is a defeaters tree for $\langle A, h\rangle$ with root $\langle A, h\rangle$.
ii. An argument structure $\langle A, h\rangle$ with defeaters $\langle A_1, h_1\rangle, \langle A_2, h_2\rangle, \ldots, \langle A_n, h_n\rangle$ is a defeaters tree with root $\langle A, h\rangle$ and its children nodes are the defeaters trees for $\langle A_1, h_1\rangle$, $\langle A_2, h_2\rangle$, ..., $\langle A_n, h_n\rangle$.

Definition **5.2** Let T_D be a defeaters tree for an argument structure $\langle A, h\rangle$. Its nodes can be labeled as follows:
i. Leaves of a T_D are *undefeated-nodes*.
ii. An inner node (including the root) is:
-*Defeated-node* if and only if it has at least a child that is an undefeated-node.
-*Undefeated-node* if and only if all its children are defeated-nodes.

Definition **5.3** Let $\langle A, h\rangle$ be an argument structure for h. We say that T_I is an *interference tree* if T_I is a defeaters tree for $\langle A, h\rangle$ and its root is an argument structure $\langle S, r\rangle$ that is a counterargument for $\langle A, h\rangle$ and it is labeled as undefeated-node.

Definition **5.4** Let $\langle A, h\rangle$ be an argument structure for h. We say that $\langle A, h\rangle$ is a *justification* for h if there is no interference tree T_I for $\langle A, h\rangle$.

From these definitions, when the system tries to justify h, there will be four possible answers: "yes", "unknown", "no" and "undefined". The answer will be "yes" if there exists a justification for h. The system will answer "unknown" if there is no argument structure $\langle A, h\rangle$. The answer will be "no" if every argument structure $\langle A, h\rangle$ has at least one interference tree T_I whose root $\langle S, r\rangle$ is a defeater for $\langle A, h\rangle$. Otherwise, the answer will be "undefined".

6. THE ARGUMENTS MAINTENANCE SYSTEM

The reason for introducing the Arguments Base $\mathbb{B}$ is to save work already done when looking for a justification. If no new facts are added to $\mathcal{K}$, some queries could be answered just by looking in $\mathbb{B}$ without having to recurse to the inference mechanism. On the other hand, it is desirable that a system modelling the behavior of an intelligent agent will have the capability of internalizing new information dynamically. The system provides this service, along with the capability of adding particular facts (grounded literals) to the knowledge base $\mathcal{K}$. The Arguments Base $\mathbb{B}$ could be affected when new facts are added to $\mathcal{K}$: new argument structures could be generated and some arguments in $\mathbb{B}$ would become invalid. In order to keep the contents of $\mathbb{B}$ updated, the Arguments Maintenance System (AMS) will revise $\mathbb{B}$ automatically every time a new fact is introduced.

6.1. Invalidation of Arguments stored in $\mathbb{B}$

Adding consistently a particular fact f to $\mathcal{K}$ could render invalid some of the arguments stored in $\mathbb{B}$. Looking at definition 2.1 we see that condition (1) will remain valid no matter what we add to $\mathcal{K}$. The situation with respect to the consistency and minimality of the arguments is clearly different.

We will first analyze the consistency condition. Let $\mathcal{K}'$ be the expansion of $\mathcal{K}$ by f, *i.e.*, $\mathcal{K}' = \mathcal{K} \cup \{f\}$. An argument $A = \{R_1, R_2, \ldots, R_n\} \in \mathbb{B}$ is consistent with respect to $\mathcal{K}'$ (that is, $\mathcal{K}' \cup A \not\mathrel{|\!\!\sim} \perp$), if for every grounded defeasible rule

$R_i = a_1 \wedge a_2 \wedge \ldots \wedge a_k \mathrel{>\!\!\!-} c_i \in A$ holds $\mathcal{K}' \cup \{R_1, \ldots, R_{i-1}\} \not\mathrel{|\!\!\!\sim} \neg c_i$ for $1 \quad i \quad n$. If A is consistent with $\mathcal{K}'$, then A remains in $\mathbb{B}$, otherwise it is discarded.

Let $\langle A, h\rangle$ be an argument structure. Minimality of $\langle A, h\rangle$ could be violated when the new fact allows the construction of a new argument A' for h such that $A' \subset A$. If no consequent c of any defeasible rule R_i in A is such that $\mathcal{K} \cup \{f\} \vdash c$, then A remains minimal. On the other hand, if $\mathcal{K} \cup \{f\} \vdash c$ for some rule R_i in A, then R_i is a redundant rule in A, and can be eliminated. After eliminating all redundant rules from A, a minimal argument A' is obtained. Finally, A will be replaced by A' in $\mathbb{B}$.

6.2. Generating new arguments

We also need to update $\mathbb{B}$ when the addition of a new fact allows the construction of new arguments. Let $\mathcal{K}' = \mathcal{K} \cup \{f\}$ be the expanded knowledge base by the addition of f. In order to generate the new argument structures, the AMS uses a combined method of *forward–chaining* along with the defeasible inference *backward–chaining* mechanism. The method used for updating $\mathbb{B}$ is the following: the addition of a new fact f [7] to $\mathcal{K}$ could permit the firing of some rule R (weak or strong) that could not be fired from $\mathcal{K}$ alone. The rule R will be fired if f unifies with one of the literals in the antecedent of R, and the remaining literals have a DIT (obtained by backward-chaining). Thus, the AMS obtains a new argument structure $\langle A, h\rangle$, where h is the consequent of R. The literal h could also unify with some other literal in the antecedent of another rule R'. Then, a new argument for the consequent of R' can be obtained. The process will continue until all new arguments the new fact has made possible have been generated.

Example **6.1** Consider the following knowledge base:

$$\mathcal{K} = \{\ bird(petete),\ penguin(X) \twoheadrightarrow \neg flies(X)\ \}$$

$$\begin{aligned}\Delta = \{\ & bird(X) \mathrel{>\!\!\!-} flies(X),\\ & penguin(X) \mathrel{>\!\!\!-} swims(X),\\ & bird(X) \mathrel{>\!\!\!-} lives\_on\_land(X),\\ & lives\_on\_land(X) \wedge swims(X) \mathrel{>\!\!\!-} lives\_near\_water(X)\ \}\end{aligned}$$

and an Arguments Base $\mathbb{B}$ containing

$$\langle \{bird(petete) \mathrel{>\!\!\!-} flies(petete)\}, flies(petete)\rangle$$
$$\langle \{bird(petete) \mathrel{>\!\!\!-} lives\_on\_land(petete)\}, lives\_on\_land(petete)\rangle$$

After adding the fact $penguin(petete)$ to $\mathcal{K}$, the following steps are taken:

1. The argument $\langle \{bird(petete) \mathrel{>\!\!\!-} flies(petete)\}, flies(petete)\rangle$ becomes invalid, since $flies(petete)$ is no longer consistent with the knowledge base $\mathcal{K}$.

2. The rule $penguin(X) \mathrel{>\!\!\!-} swims(X)$ is used to build the argument $\langle\ \{\ penguin(petete) \mathrel{>\!\!\!-} swims(petete)\ \},\ swims(petete)\ \rangle$, and $swims(petete)$ becomes a new fact to be considered in the forward–chaining process.

3. The fact $swims(petete)$ unifies with one literal in the antecedent of the rule $lives\_on\_land(X) \wedge swims(X) \mathrel{>\!\!\!-} lives\_near\_water(X)$. Since $lives\_on\_land(petete)$ has a DIT, this rule will be fired, allowing the generation of the argument $\langle\ \{\ penguin(petete) \mathrel{>\!\!\!-} swims(petete),\ lives\_on\_land(petete) \wedge swims(petete) \mathrel{>\!\!\!-} lives\_near\_water(petete)\ \},\ lives\_near\_water(petete)\ \rangle$.

[7] With "new fact f", we mean "a grounded literal f such that $\mathcal{K} \not\vdash f$"

Let $N = \{\langle N_1, h_1\rangle, \ldots, \langle N_k, h_k\rangle\}$ be the set of the newly formed argument structures created after the addition of f to $\mathcal{K}$. For each N_i in N, the AMS will find out if N_i counterargues any member $\langle A, h\rangle$ in $\mathbb{B}$. The appropriate action will be taken, updating the information associated to $\langle A, h\rangle$ and N_i.

There are two special cases in which the addition of f to $\mathcal{K}$ does not affect $\mathbb{B}$. These are: (i) f is an instance of a literal where the combination of the predicate letter and arity does not appear among the literals of $\mathcal{K} \cup \Delta$, and (ii) f is an instance of a literal such that $\mathcal{K} \vdash f$.

7. CONCLUSIONS

The way from a solid theoretical foundation to efficient argument based systems promises to be full of interesting aspects. We have introduced some conceptualizations in terms of trees (inference trees, defeaters trees, *etc.*) obtaining an easier way of specifying our algorithms. The updating of an Arguments Base after the addition of a new fact to the knowledge base is the first step in the direction of a system that would allow to update the knowledge base (facts and strong rules) and the set of defeasible (weak) rules. Finally, Argument Based Systems show the possibility of developing Knowledge Based Systems beyond Rule Based Systems.

8. REFERENCES

García, A.J., Chesñevar,C.I. and Simari,G.R., 1993, Bases de argumentos: su mantenimiento y revisión, in *XIX Conferencia Latinoamericana de Informática, 22as. Jornadas Argentinas de Informática e Investigación Operativa.*

Lloyd,G., 1987, Foundations of Logic Programming, Springer-Verlag, 2nd. Edition.

Loveland,D., 1978, Automated Theorem Proving: A Logical Basis, North Holland.

McCarthy,J.,1980, Circunscription – A form of non-monotonic reasoning, *Artificial Intelligence* 13: 27–39.

McDermott,D. and Doyle,J., 1980, Non-monotonic logic I, *Artificial Intelligence*, 13: 41–72.

Lin,F. and Shoham,Y.,1989, Argument systems: a uniform basis for nonmonotonic reasoning, STAN-CS-89-1243, Stanford University, Department of Computer Science.

Moore,R.C.,1985, Semantical considerations on nonmonotonic logic, in *Artificial Intelligence*, 25:(1) 75–94.

Pollock,J.L., 1987, Defeasible reasoning, in *Cognitive Science*, 11:481–518.

Poole,D.L., 1985a, On the comparison of theories: preferring the most specific explanation, in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 144–147, IJCAI.

Poole,D.L., Aleliunas,R. and Goebel,R., 1985b, THEORIST: A logical reasoning system for defaults and diagnosis, Technical Report, Departament of Computer Science, University of Waterloo, Waterloo, Canada.

Poole,D.L., 1988, A logical framework for default reasoning, in *Artificial Intelligence* 36, pp. 27–47.

Prakken,H.,1993, Logical Tools for Modelling Legal Arguments, PhD Thesis, Vrije University, Amsterdam, Holland.

Reiter,R., 1980, A logic for default reasoning, in *Artificial Intelligence*, 13: 81–132.

Simari,G.R., and Loui,R.P., 1992, A mathematical treatment of defeasible reasoning and its implementation, in *Artificial Intelligence*, 53: 125–157.

Vreeswijk,G.,1991, On the feasibility of defeasible reasoning, in *Knowledge Representation '91*.