Report Number: WUCS-89-12

1989-09-01

A Mathematical Treatment of Defeasible Reasoning and its Implementation

Authors: Guillermo R. Simari and R. P. Loui

We present a mathematical approach to defeasible reasoning. This approach is based on the notion of specificity introduced by Poole and the theory of warrant presented by Pollock. We combine the ideas of the two. This main contribution of this paper is a precise well-defined system which exhibits correct behavior when applied to the benchmark examples in the literature.
We prove that an order relation can be introduced among equivalence classes under the equi-specificity relation. We also prove a theorem that ensures the termination of the process of finding the justified facts. Two more lemmas define a reduced search space for checking specificity.
In order to implement the theoretical ideas, the language is restricted to Horn clauses for the evidential context. The language used to represent defeasible rules has been restricted in a similar way. The authors intend this work to unify the various existing approaches to argument-based defeasible reasoning.
... **Read complete abstract on page 2.**

A Mathematical Treatment of Defeasible Reasoning and its Implementation

Complete Abstract:

We present a mathematical approach to defeasible reasoning. This approach is based on the notion of specificity introduced by Poole and the theory of warrant presented by Pollock. We combine the ideas of the two. This main contribution of this paper is a precise well-defined system which exhibits correct behavior when applied to the benchmark examples in the literature.

We prove that an order relation can be introduced among equivalence classes under the equi-specificity relation. We also prove a theorem that ensures the termination of the process of finding the justified facts. Two more lemmas define a reduced search space for checking specificity.

In order to implement the theoretical ideas, the language is restricted to Horn clauses for the evidential context. The language used to represent defeasible rules has been restricted in a similar way. The authors intend this work to unify the various existing approaches to argument-based defeasible reasoning.

A MATHEMATICAL TREATMENT OF DEFEASIBLE REASONING AND ITS IMPLEMENTATION

Guillermo R. Simari
Ronald P. Loui

WUCS-89-12

September 1989

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

Submitted to the AI Journal, January 1990.

Abstract

We present a mathematical approach to defeasible reasoning. This approach is based on the notion of specificity introduced by Poole and the theory of warrant presented by Pollock. We combine the ideas of the two. The main contribution of this paper is a precise, well-defined system which exhibits correct behavior when applied to the benchmark examples in the literature.

We prove that an order relation can be introduced among equivalence classes under the equi-specificity relation. We also prove a theorem that ensures the termination of the process of finding the justified facts. Two more lemmas define a reduced search space for checking specificity.

In order to implement the theoretical ideas, the language is restricted to Horn clauses for the evidential context. The language used to represent defeasible rules has been restricted in a similar way.

The authors intend this work to unify the various existing approaches to argument-based defeasible reasoning.

1 Introduction

Recent courage to deviate from standard practice in non-monotonic reasoning has led to an influx of formalisms. Each achieves non-monotonicity in a first-order language where entailment is not based on fixed points, nor on model minimization. Most avoid intensional contexts by semantic ascent,[1] thus supplementing the proof theory in the metalanguage. This obviates the need for model-theoretic accounts of new syntax, since there is no new syntax.

Inspiration has come from conditional logics (Nute[16, 15], Delgrande[2], Glymour-Thomason[6]) or inductive logics. In the latter case, both induction's form (Loui[10], Pollock[18]) and its effect (Geffner-Pearl[5], Neufeld[14]) have been copied. All of the resulting systems incorporate a specificity defeater, analogous to the subclass defeater in inheritance with exceptions (since Touretzky[26]).

Some of these authors have found use for objects called *arguments* (also *theories*). Other systems are based on irrelevance. This paper is concerned with those based on arguments.

Arguments are *prima facie* proofs that may make use of the non-monotonic relations. They indicate support for a proposition, but do not establish warrant once and for all; it matters what other counterarguments there may be. Arguments may have structure (Loui[10], Pollock[18]) or may just be collections of supporting sentences (Poole[20], Geffner-Pearl[5]). There is widespread agreement that arguments in these systems generalize paths in inheritance systems.[2]

[1] Quine's phrase, in private communication.

[2] General discussion during the Workshop on Defeasible Reasoning with Specificity and Multiple Inheritance, St. Louis, April, 1989.

As is the case in inheritance, there is a "clash of intuitions" that has resulted in a plethora of theories. There are at present few ways of classifying the systems. Our intent, in defining yet another system, is not to add to the inventory. In fact, this paper attempts to bring together the prominent systems based on arguments. A system is defined that takes its form from Loui (which in turn evolved from Kyburg[7]) and which combines the rules of Poole and of Pollock. For most of the AI audience, this will effectively condense three systems into one, remedying deficiencies of each.

More importantly, this system is defined in a mathematically more rigorous manner. Past definitions (especially Poole and Loui) did not have the precision nor the completeness to serve as a foundation for future mathematical work. It is no accident that the statement of the system here allows concise proof of non-trivial properties.

1.1 Poole and Pollock Combined

Poole treats specificity, *i.e.* a comparative measure of the relevance of information, in the right way but stops short in the operational aspects of his system. Thus, it is impossible to decide when to apply his *specificity comparator*. On the other hand, Pollock treats the operational aspects correctly but he rejects specificity as a generalization of the subclass defeater, even as a useful shorthand. This rejection places him in an extreme minority in the defeasible reasoning community.

In our view, Poole and Pollock fail to pursue the theoretic ideas in their systems. Poole [19] has implemented a system of defeasible reasoning which does not address the complexity of interactions among arguments' specificity. Pollock has taken his research in the statistical direction, which is too general.

The system defined here combines the ideas of the two. But the main contribution of this paper is a precise, well-defined system which exhibits a correct behavior when applied to the benchmark examples in the literature.[3]

We take the knowledge of agent a to be divided into a set of defeasible rules Δ and a set of well-formed formulas (*wffs*) $\mathcal{K}$ in the standard formal logic sense. The set $\mathcal{K}$ is further divided in grounded wffs, the *contingent* part of $\mathcal{K}$, and ungrounded wffs, the *necessary* part of $\mathcal{K}$. Evidence suggests new tentative conclusions. Now any new tentative conclusion p will be *judged* in light of being consistent with $\mathcal{K}$ and having a supporting subset of Δ which, in conjunction with $\mathcal{K}$, can derive p without deriving a contradiction at the same time.

The operational part of accepting or rejecting p in the judgment above, can be understood as a dialectic in the following sense. As we said before, p must be consistent with $\mathcal{K}$, but its interaction with subsets of Δ could be more interesting. Together with $\mathcal{K}$, Δ may contain certain subsets which support p and subsets which support its negation. If a subset S of Δ supports p, we will say that there exists a defeasible derivation of p from S. The process should "weigh" those subsets against each other and decide which one of the expressions of p can be supported. The comparison of the subsets, called *argument structures*, is done based on the specificity relation which looks at the set of closed sentences in the language that, together with the argument structure and the necessary part of the knowledge, are enough to produce a defeasible derivation. The supported conclusion will be given by the winning argument structure, if one exists. We will formally describe this process next.

[3] Actual solution of two dozen such examples can be obtained from the authors.

2 Arguments and Specificity

We will construct a formal system IL with the objective of providing a language in which to represent the knowledge of a given agent a and in which to perform his defeasible reasoning. Our purpose is to create a framework in which the agent will be able to formulate tentative arguments.

These arguments, represented in IL's formal language, will be the subject of a dialectical process which will establish a preference order on them. The order will be partial because obviously arguments may be unrelated. The dialectical process will compare the arguments looking for other arguments that could impair the applicability on argument's basis. Finally, when those *counterarguments* are found, they will be *compared* with the original argument using the preference partial order. Poole [20] has suggested a definition for the partial order and we will further explore the implications of that definition. Pollock [18] has given a characterization of the process through which arguments become justified. That characterization will be expanded and reformulated in our framework.

The language of IL is composed of a *first order language* $\mathcal{L}$, plus a binary meta-linguistic relation among members of $\mathcal{L}$. Any axiomatization of $\mathcal{L}$ will do for our purposes, and we will use the standard *connectives* and *punctuation symbols* [12, 3] freely without explicitly introducing them. We assume that the rules of inference attached to the axiomatization are *modus ponens* and *Generalization*. The members of the meta-linguistic relation are called *defeasible rules* and they have the form $\alpha \succ\!\!- \beta$, where α and β are well-formed formulas (*wffs*) in $\mathcal{L}$, which must contain free variables, *e.g.* they are non-closed wffs. The relation "$\succ\!\!-$" among $\mathcal{L}$ wffs is understood as expressing that "reasons to believe in the antecedent

α provide reasons to believe in the consequent β". Variables with the same name on both sides of the rule are assumed to be the same. An instance of an open defeasible rule is obtained by replacing *all* the free variables by appropriate constants. These rules enable us to represent the uncertain nature of arguments. When no confusion is possible we will use the term defeasible rule to refer to the open defeasible rule and to its grounded instances.

The set $Sent(\mathcal{L})$ of *sentences* of $\mathcal{L}$, that is the set of closed well-formed formulas in $\mathcal{L}$, can be partitioned in two subsets, corresponding to necessary and contingent information. Necessary information is the context in which defeasible rules are provided. Although a purely syntactic distinction might not be possible on philosophical grounds, we normally take sentences with variables to be necessary. Thus, the first subset contains the grounded sentences $Sent_{\mathcal{C}}(\mathcal{L})$ and the second subset contains non-grounded sentences $Sent_{\mathcal{N}}(\mathcal{L})$, *i.e.* $Sent(\mathcal{L}) = Sent_{\mathcal{C}}(\mathcal{L}) \cup Sent_{\mathcal{N}}(\mathcal{L})$. Obviously, $Sent_{\mathcal{C}}(\mathcal{L}) \cap Sent_{\mathcal{N}}(\mathcal{L}) = \emptyset$. The names used for the subsets reflect the view that the grounded sentences in $Sent_{\mathcal{C}}(\mathcal{L})$ represent information depending on the individual constants of the language. Those individual constants are *contingent* to the reality being represented. On the other hand, the sentences in $Sent_{\mathcal{N}}(\mathcal{L})$ are wffs containing variables. That characteristic allows them to convey properties that single out a *class* of worlds, *i.e.* worlds where the relations among individuals are the same regardless of the *local* individuals. We choose to call these sentences the *necessary* facts, because without them the world wouldn't be as it is.

The knowledge of a is represented in $\mathbb{L}$ by a pair $(\mathcal{K}, \Delta)$, where $\mathcal{K}$ is a subset of $Sent(\mathcal{L})$, and Δ is a finite set of defeasible rules. The pair $(\mathcal{K}, \Delta)$ will be called a *Defeasible Logic Structure*. $\mathcal{K}$ represents the indefeasible part of a's knowledge and Δ represents tentative information, *i.e.* information that a is prepared to take at less than face value. In mapping

a's reality to a subset $\mathcal{K}$ of $\mathcal{L}$ we obtain a partition of $\mathcal{K}$ in two subsets $\mathcal{K}_{\mathcal{N}} = Sent_{\mathcal{N}}(\mathcal{L}) \cap \mathcal{K}$ and $\mathcal{K}_{\mathcal{C}} = Sent_{\mathcal{C}}(\mathcal{L}) \cap \mathcal{K}$. Clearly, $\mathcal{K} = \mathcal{K}_{\mathcal{N}} \cup \mathcal{K}_{\mathcal{C}}$. The only condition on $\mathcal{K}$ is consistency, *i.e.* $\mathcal{K} \not\vdash \bot$. Sometimes, when using $\mathcal{K}$, we will refer to it as the *context*, and $\mathcal{K}$ will be considered as a set of wffs or as the conjunction of them depending on the situation.

Having defined our knowledge representation language we need to introduce a notion of entailment, or inference, which is somewhat different from the one used in first order languages. That is, given a Defeasible Logic Structure ($\mathcal{K}$,Δ), we need to define what other facts can be sanctioned as *justified.* Our formal system introduces this notion in a way that is not axiomatic. For a complete definition we need to further develop our formalism. We will present the syntactic part here. The rest will be introduced in the next sections.

Given a member A of $Sent(\mathcal{L})$, and set $\Gamma = \{A_1, A_2, \ldots, A_n\}$, where each A_i is a member of $\mathcal{K}$ or a grounded instance of a member of Δ, we will establish a meta-meta-relationship "$\mid\!\sim$", called *defeasible consequence*, between Γ and A in the following way. A well-formed formula A will be called a defeasible consequence of the set Γ as described above, if and only if there exists a sequence $B_1, \ldots, B_m$ such that $A = B_m$ and for each i, either A_i is an axiom of $\mathcal{L}$ or A_i is in Γ, or A_i is a direct consequence of the preceding members of the sequence using modus ponens or instantiation of a universally quantified sentence. The grounded instances of the defeasible rules are regarded as material implications for the application of modus ponens. The sequence $B_1, \ldots, B_m$ will be called a *defeasible derivation* or just a *derivation.* We use $\Gamma \mid\!\sim A$ as an abbreviation of *A is a defeasible consequence of Γ*. If necessary, in order to avoid confusion with the context, we write $\Gamma \mid\!\sim_{\mathcal{K}} A$. We also will write $A_1, \ldots, A_n \mid\!\sim A$ instead of $\{A_1, \ldots, A_n\} \mid\!\sim A$, and $\mathcal{K} \cup T \mid\!\sim A$, making explicit the distinction between the context $\mathcal{K}$ and a set T of defeasible rules used in the derivation.

In first order logic the above definition is enough to describe the wffs that are theorems of Γ, but for us the situation is different because we need to introduce the tentative nature of the conclusions, *e.g.* we need to give a criterion that will allow us to prefer one conclusion over another. That criterion will be the specificity relation among arguments. We will now introduce the formal notion of arguments and later we will define the specificity relation among those formal objects.

2.1 Arguments

Derivations, as defined above, make use of some grounded instances of defeasible rules from Δ. The set of grounded defeasible rules characterize the derivation and we will give the name *argument basis* to that set. In order to facilitate the following discussion we introduce the set $\Delta^{\downarrow}$ of all grounded instances of members of Δ produced by using the individual constants in $\mathcal{L}$.

DEFINITION 2.1 [*Preliminary*] Given a context $\mathcal{K} = \mathcal{K}_{\mathcal{N}} \cup \mathcal{K}_{\mathcal{C}}$ and a set Δ of defeasible rules we say that a subset T of $\Delta^{\downarrow}$, is an *argument* for $h \in Sent_{\mathcal{C}}(\mathcal{L})$ in the context $\mathcal{K}$, denoted by $\langle T, h\rangle_{\mathcal{K}}$, if and only if

$$\mathcal{K} \cup T \mid\sim h$$

$$\mathcal{K} \cup T \not\mid\sim \bot$$

The pair $\langle T, h\rangle_{\mathcal{K}}$ will be called an *argument structure.*

REMARKS: When possible we will drop the reference to the context and we will write $\langle T, h\rangle$ meaning $\langle T, h\rangle_{\mathcal{K}}$. We will refer to the collection of all possible argument structures as $\mathbf{AStruc}(\Delta^{\downarrow})$ or just $\mathbf{AStruc}$. There is a distinguished argument, for any context $\mathcal{K}$, $\langle \emptyset, \mathcal{K}^{\vdash}\rangle$, *i.e.* no rules are necessary to support the conjunction of the atoms of the deductive closure ($\mathcal{K}^{\vdash}$) of the knowledge in $\mathcal{K}$. Finally, for $\langle T, h\rangle$ we will assume that the set T is minimal, or non-redundant in the sense that it does not contain any rule that is unnecessary for the inference of h. This is a sort of "Occam's Razor" principle for arguments.

DEFINITION 2.2 [*Revised*] Given a context $\mathcal{K} = \mathcal{K}_{\mathcal{N}} \cup \mathcal{K}_{\mathcal{C}}$ and a set Δ of defeasible rules we say that a subset T of $\Delta^{\downarrow}$, is an *argument* for $h \in Sent_{\mathcal{C}}(\mathcal{L})$ in the context $\mathcal{K}$, denoted by $\langle T, h\rangle_{\mathcal{K}}$, if and only if

1. $\mathcal{K} \cup T \mid\sim h$
2. $\mathcal{K} \cup T \not\vdash \bot$
3. $\nexists T' \subset T,\ \mathcal{K} \cup T' \mid\sim h$

EXAMPLE 2.1

Let $\mathcal{K}$= $\{P(a), Q(a)\}$ and Δ= $\{P(x)\succ\!\!-R(x), Q(x) \wedge R(x)\succ\!\!-H(x),\ M(x)\succ\!\!-N(x)\}$ be the context and defeasible rule set respectively. Therefore the subset T of grounded instances of defeasible rules in Δ, $\{P(a)\succ\!\!-R(a), Q(a) \wedge R(a)\succ\!\!-H(a)\}$ is an argument structure for $H(a)$, *i.e.* $\langle T, H(a)\rangle$ is an argument structure.

DEFINITION 2.3 Let $\langle T, h\rangle$ be an argument structure for h, and $\langle S, j\rangle$ an argument structure for j such that $S \subseteq T$. We will say that $\langle S, j\rangle$ is a *subargument* of $\langle T, h\rangle$ and use the notation $\langle S, j\rangle \subseteq \langle T, h\rangle$, overloading the symbol "$\subseteq$".

EXAMPLE 2.2 Given any argument structure $\langle T, h\rangle$, the two argument structures $\langle \emptyset, \mathcal{K}^{\vdash}\rangle$ and $\langle T, h\rangle$ are two trivial subarguments of it.

EXAMPLE 2.3 In the conditions of the above example $S_1 = \{P(a)\succ\!\!-R(a), Q(a) \wedge R(a)\succ\!\!-H(a)\}$ is an argument for $H(a)$, and $S_2 = \{P(a)\succ\!\!-R(a)\}$ is an argument for $R(a)$. We have the following relation among the argument structures $\langle S_1, H(a)\rangle \subseteq \langle S_1, H(a)\rangle$ and $\langle S_2, R(a)\rangle \subseteq \langle S_1, H(a)\rangle$.

Sometimes it will be necessary to talk about the defeasible rules in terms of their antecedents and consequents. The following definitions introduce three operators for this purpose.

DEFINITION 2.4 Let T be a subset of Δ^1. We will introduce two operators over sets of defeasible rules. They are the operator $An(\cdot)$, which applied to T will return the set of antecedents of its rules, and $Co(\cdot)$, which applied to T will return the set of consequents of its rules.

EXAMPLE 2.4 Given the argument $T = \{A(r) \succ\!\!- D(r), B(r) \wedge D(r) \succ\!\!- C(r), C(r) \succ\!\!- E(r)\}$ then we have $An(T) = \{A(r), B(r), D(r)\}$ and $Co(T) = \{D(r), C(r), E(r)\}$.

It is also useful to have access to the set of literals used in the defeasible rules of an argument structure.

DEFINITION 2.5 Let $\langle T, h\rangle$ be an argument structure. Then the operator $Lit(\cdot)$ will return the set of literals in T with the exception of h, *i.e.* $Lit(\langle T, h\rangle) = An(T) \cup Co(T) - \{h\}$.

EXAMPLE 2.5 Given the argument $T = \{A(r) \succ\!\!- D(r), B(r) \wedge D(r) \succ\!\!- C(r), C(r) \succ\!\!- E(r)\}$ then we have $Lit(T) = \{A(r), B(r), C(r), D(r)\}$.

2.2 Specificity

Having defined these objects we would like to establish certain binary relations on $\mathrm{AStruc}(\Delta^t)$ in such a way that it would help us to choose the "better" argument structure that supports a conclusion. The following definitions, essentially Poole's [20], will characterize this relation.

DEFINITION 2.6 Given two argument structures $\langle T_1, h_1\rangle$, $\langle T_2, h_2\rangle$ in AStruc, we say that T_1 for h_1 is *strictly more specific than* T_2 for h_2 denoted by

$$\langle T_1, h_1\rangle \succ_{\mathbf{spec}} \langle T_2, h_2\rangle,$$

if and only if

1. $\forall e \in Sent_{\mathcal{C}}(\mathcal{L})$ such that $\mathcal{K}_{\mathcal{N}} \cup \{e\} \cup T_1 \mid\sim h_1$ also $\mathcal{K}_{\mathcal{N}} \cup \{e\} \cup T_2 \mid\sim h_2$, and

2. $\exists e \in Sent_{\mathcal{C}}(\mathcal{L})$ such that:

$$\mathcal{K}_{\mathcal{N}} \cup \{e\} \cup T_2 \mid\sim h_2 \qquad \textit{(Activates } T_2\textit{)},$$

$$\mathcal{K}_{\mathcal{N}} \cup \{e\} \cup T_1 \not\mid\sim h_1 \qquad \textit{(Does not activate } T_1\textit{)}$$

$$\mathcal{K}_{\mathcal{N}} \cup \{e\} \not\mid h_2 \qquad \textit{(Non-triviality condition)}$$

REMARK: The term *activates* appearing in the definition is used with the following meaning: *together with $\mathcal{K}_{\mathcal{N}}$ and the argument T is enough to construct a defeasible derivation of h.*

Another important relation among argument structures is the notion of being equally specific.

DEFINITION 2.7 Two argument structures T_1 for h_1 and T_2 for h_2 are *equi-specific*, denoted by

$$\langle T_1, h_1\rangle \approx_{\text{spec}} \langle T_2, h_2\rangle,$$

when the following condition holds,

$$\forall e \in Sent_{\mathcal{C}}(\mathcal{L}) \text{ , } \mathcal{K}_{\mathcal{N}} \cup \{e\} \cup T_1 \mid\sim h_1 \text{ if and only if } \mathcal{K}_{\mathcal{N}} \cup \{e\} \cup T_2 \mid\sim h_2.$$

Finally the combination of both notions gives the following definition.

DEFINITION 2.8 We say that an argument structure T_1 for h_1 is *at least as specific as* an argument structure T_2 for h_2 denoted by $\langle T_1, h_1\rangle \succeq_{\text{spec}} \langle T_2, h_2\rangle$, if and only if $\langle T_2, h_2\rangle \approx_{\text{spec}} \langle T_1, h_1\rangle$ or $\langle T_1, h_1\rangle \succ_{\text{spec}} \langle T_2, h_2\rangle$.

Some examples will clarify the concept.

EXAMPLE 2.6 The argument structure $\langle \{A(r) \wedge B(r) \!>\!\!\!-\!\!\!- C(r)\}, C(r)\rangle$ is more specific than $\langle \{A(r) \!>\!\!\!-\!\!\!- \neg C(r)\}, \neg C(r)\rangle$ because every time the first argument can be activated to support $C(r)$ the second also supports $\neg C$. But, on the other hand, $A(r)$ alone can activate the second argument structure but does not activate the first.

So $\langle \{A(r) \wedge B(r) \!>\!\!\!-\!\!\!- C(r)\}, C(r)\rangle \succ_{\text{spec}} \langle \{A(r) \!>\!\!\!-\!\!\!- \neg C(r), \neg C(r)\rangle$

EXAMPLE 2.7 The argument structure $\langle \{A(r) \!>\!\!\!-\!\!\!- \neg C(r)\}, \neg C(r)\rangle$ is more specific than the argument structure $\langle \{A(r) \!>\!\!\!-\!\!\!- B(r), B(r) \!>\!\!\!-\!\!\!- C(r)\}, C(r)\rangle$ because every time the first argument can support $\neg C(r)$ the second also supports $C(r)$. But, on the other hand, $B(r)$ alone can activate the second argument structure but does not activate the first.

So $\langle \{A(r) \!>\!\!\!-\!\!\!- \neg C(r)\}, \neg C(r)\rangle \succ_{\text{spec}} \langle \{A(r) \!>\!\!\!-\!\!\!- B(r), B(r) \!>\!\!\!-\!\!\!- C(r)\}, C(r)\rangle$

REMARK: Whenever no confusion is possible we will drop the subscript "spec" in the symbols "$\succ_{\text{spec}}$","$\succeq_{\text{spec}}$" and "$\approx_{\text{spec}}$" writing instead "$\succ$", "$\succeq$" and "$\approx$".

An argument and its subarguments are related by the specificity relation in a natural, expected way.

LEMMA 2.1 *Let $\langle T, h\rangle$ be an argument structure and $\langle S, j\rangle$ a subargument of $\langle T, h\rangle$. Then $\langle T, h\rangle$ is more specific than $\langle S, j\rangle$, i.e. $\langle T, h\rangle \succeq \langle S, j\rangle$.*

The equi-specificity "$\approx$" relation decomposes AStruc into disjunct subsets of equi-specific arguments, *i.e.* establishes a partition on it. This property is better expressed in the following lemma.

LEMMA 2.2 *The equi-specificity relation among members of* AStruc *is an equivalence relation.*

The "$\approx$" equivalence relation will help us to introduce an order relation in the set $\text{AStruc}/_{\approx}$, *i.e.* the quotient set of AStruc by the equivalence relation "$\approx$". This order relation is induced by "$\succeq$". First, we observe that $\succeq$ defines a quasi-ordering in AStruc, *i.e.* the relation is reflexive and transitive (see [4]). If we lift the relation to the quotient set $\text{AStruc}/_{\approx}$ of the equivalence classes defined by "$\approx$" in AStruc the new relation will define a partial order over those classes, as is shown in the lemma below.

REMARK: For all $\langle T, h\rangle$ in AStruc the notation $[\langle T, h\rangle]$ represents the equivalence class of $\langle T, h\rangle$ in $\text{AStruc}/_{\approx}$.

DEFINITION 2.9 We define the relation "$\sqsupseteq_{\text{spec}}$" in the quotient set $\text{AStruc}/_{\approx}$ as follows. Given $[\langle T_1, h_1\rangle], [\langle T_2, h_2\rangle]$ in $\text{AStruc}/_{\approx}$ then

$$[\langle T_1, h_1\rangle] \sqsupseteq [\langle T_2, h_2\rangle] \text{ if and only if } \langle T_1, h_1\rangle \succeq \langle T_2, h_2\rangle \ .$$

Again, whenever possible we will drop the "spec" subscript from "$\sqsupseteq_{\text{spec}}$" writing "$\sqsupseteq$".

The introduction of the $\succeq$ relation on AStruc has the objective of providing a way to select the most "appropriate" argument structure. In that sense the following lemma establishes the fundamental property regarding order in $\text{AStruc}/_{\approx}$.

LEMMA 2.3 *The relation* $\sqsupseteq$ *defined in* $\text{AStruc}/_{\approx}$ *is a partial order.*

The next lemma defines a reduced search space for checking specificity.

LEMMA 2.4 *Let* $\langle T_1, h_1\rangle$ *and* $\langle T_2, h_2\rangle$ *be two argument structures in* AStruc. *Then the following conditions are equivalent:*

1. $\langle T_1, h_1\rangle \succeq \langle T_2, h_2\rangle$
2. $\forall x \in \text{An}(T_2),\ \mathcal{K}_{\mathcal{N}} \cup \text{An}(T_1) \cup T_2 \mid\!\sim x$

Proof :

1 implies 2)

Assume that $\langle T_1, h_1\rangle \succeq \langle T_2, h_2\rangle$. Hence, $\langle T_1, h_1\rangle$ is at least as specific as any subargument of $\langle T_2, h_2\rangle$. There is always a subargument of $\langle T_2, h_2\rangle$ for any x in $An(T_2)$, (by the non-redundant property of $\langle T_2, h_2\rangle$). Therefore, $\mathcal{K}_{\mathcal{N}} \cup An(T_1) \cup T_2 \mid\!\sim x$ for all $x \in An(T_2)$.

2 implies 1)

Assume that e in $Sent_{\mathcal{C}}(\mathcal{L})$ is such that $\mathcal{K}_{\mathcal{N}} \cup \{e\} \cup T_1 \mid\!\sim h_1$. We want to show that $\mathcal{K}_{\mathcal{N}} \cup \{e\} \cup T_2 \mid\!\sim h_2$. Because of (2) every $An(T_2)$ can be derived, therefore every $Co(T_2)$ is

defeasibly derived; hence the $\langle T_2, h_2\rangle$ is activated, *i.e.* $\mathcal{K} \cup \{e\} \cup T_2 \mid\sim h_2$. That is, $\langle T_1, h_1\rangle \succeq \langle T_2, h_2\rangle$. □

LEMMA 2.5 *Let* $\langle T_1, h_1\rangle$ *and* $\langle T_2, h_2\rangle$ *be such that* $\langle T_1, h_1\rangle \succeq \langle T_2, h_2\rangle$. *Let* $\langle T_2, h_2\rangle$ *be such that* $\forall x \in \mathrm{Co}(T_2)$, $\mathcal{K} \cup T_1 \mid\sim x$. *If* $\langle T_2, h_2\rangle$ *contains a subargument structure* $\langle R, p\rangle$, *then* $\langle T_1, h_1\rangle$ *contains a subargument structure* $\langle S, p\rangle$ *such that* $\langle S, p\rangle \succeq \langle R, p\rangle$.

Proof : The subargument structure $\langle R, p\rangle$ of $\langle T_2, h_2\rangle$ is formed by the subset R of T_2. Given that every member in $Co(T_2)$ can be inferred using the rules in T_1 and $\mathcal{K}$, we can distinguish which rules are necessary to prove the subset $Co(R)$ of $Co(T_2)$, calling it S. We contend that $\langle S, p\rangle$ is the required subargument. Obviously, for all x in $Co(R)$, $\mathcal{K} \cup S \mid\sim x$, by its own definition. Therefore, any literal necessary to infer p from R is available in S. For the same reason $\langle S, p\rangle \succeq \langle R, p\rangle$. □

This establishes conditions for discarding arguments which reduces the search for argument defetears.

REMARK: Given two arguments $\langle T_1, h_1\rangle$ and $\langle T_2, h_2\rangle$ satisfying the conditions of the above lemma, we will say that $\langle T_1, h_1\rangle$ *covers* $\langle T_2, h_2\rangle$.

3 An Algebra of Arguments

A very good question regarding arguments is about the kind of operations that it is possible to define on them. We will devote the next few sections to consider certain operations on $\mathcal{F}(\langle T,h\rangle) = \{\langle T_i,h_i\rangle\}_{i\in I}$ the family of subarguments of an arbitrary argument structure $\langle T,h\rangle \in \mathrm{AStruc}$, where I is a set of indices, and explore some of its properties and interrelations. When no confusion is possible we will use $\mathcal{F}$ instead of $\mathcal{F}(\langle T,h\rangle)$.

A set of wffs in a first order language is consistent if and only if there is no formula for which that formula and its negation are theorems of the set. Our defeasible derivation relation is weaker than derivation in first order logic. It is possible to defeasibly derive contradictory conclusions from an arbitrary set of defeasible rules. Because of that characteristic we will introduce a weaker notion for sets of defeasible rules.

The following discussion omits proofs which can be found in the thesis.

DEFINITION 3.1 Given two argument structures $\langle T_1,h_1\rangle$, $\langle T_2,h_2\rangle$ in AStruc they will be called *concordant* if $\mathcal{K}\cup T_1\cup T_2 \not\vdash \bot$.

As could be expected, subarguments of a given argument structure have the property of being concordant with each other.

PROPOSITION 3.1 *Let $\langle T,h\rangle \in \mathrm{AStruc}$ be an arbitrary argument structure, and let $\mathcal{F}$ be the family $\{\langle T_i,h_i\rangle\}_{i\in I}$ of all $\langle T,h\rangle$ subargument structures, then the members of $\mathcal{F}$ are pairwise concordant.*

3.1 Argument Combination (join)

DEFINITION 3.2 Let $\langle T, h\rangle \in$ AStruc be an arbitrary argument structure, and let $\mathcal{F}$ be the family of all its subargument structures. Given $\langle T_1, h_1\rangle$, $\langle T_2, h_2\rangle$ in $\mathcal{F}$ we define the *combination* of them as the argument structure $\langle T_3, h_3\rangle$, where $T_3 = T_1 \cup T_2$ and $h_3 = h_1 \wedge h_2$. The operation will be denoted:

$$\langle T_3, h_3\rangle = \langle T_1, h_1\rangle \sqcup \langle T_2, h_2\rangle$$

PROPOSITION 3.2 *The combination of argument structures is a well-defined operation in* $\mathcal{F}$.

PROPOSITION 3.3 *Given two argument structures* $\langle T_1, h_1\rangle, \langle T_2, h_2\rangle \in \mathcal{F}$, *the combination* $\langle T_3, h_3\rangle = \langle T_1, h_1\rangle \sqcup \langle T_2, h_2\rangle$ *is such that* $\langle T_3, h_3\rangle \succeq \langle T_1, h_1\rangle$ *and* $\langle T_3, h_3\rangle \succeq \langle T_2, h_2\rangle$ *and* $\langle T_3, h_3\rangle$ *is the minimal (in* $\succeq$*) argument structure in* $\mathcal{F}$ *with that property which contains* $\langle T_1, h_1\rangle$ *and* $\langle T_2, h_2\rangle$ *as subarguments.*

PROPOSITION 3.4 (Associativity) *The combination of arguments in* $\mathcal{F}$ *is associative, i.e. if* $\langle T_1, h_1\rangle$,$\langle T_2, h_2\rangle$, *and* $\langle T_3, h_3\rangle$ *are subargument structures of* $\mathcal{F}$ *then*

$$(\langle T_1, h_1\rangle \sqcup \langle T_2, h_2\rangle) \sqcup \langle T_3, h_3\rangle = \langle T_1, h_1\rangle \sqcup (\langle T_2, h_2\rangle \sqcup \langle T_3, h_3\rangle)$$

PROPOSITION 3.5 (Commutativity) *The combination of arguments in* $\mathcal{F}$ *is commutative, i.e. if* $\langle T_1, h_1\rangle$,$\langle T_2, h_2\rangle$ *are subargument structures of* $\mathcal{F}$ *then*

$$\langle T_1, h_1\rangle \sqcup \langle T_2, h_2\rangle = \langle T_2, h_2\rangle \sqcup \langle T_1, h_1\rangle$$

DEFINITION 3.3 Given a subfamily $\{\langle T_{i_j}, h_{i_j}\rangle\}_{i_j \in J}$ of $\{\langle T_i, h_i\rangle\}_{i \in I}$, we define the *generalized combination* of the subargument structures in it as

$$\bigsqcup_{j\in J}\{\langle T_{i_j}, h_{i_j}\rangle\}_{i_j\in J} \;=\; \langle \bigcup_{j\in J} T_{i_j}, \bigwedge_{j\in J} h_{i_j}\rangle$$

PROPOSITION 3.6 *The argument structure* $\langle \emptyset, \mathcal{K}^{\vdash}\rangle$ *is an identity element with respect to the combining operation in the family* $\mathcal{F}$ *of subargument structures of a given argument structure* $\langle T, h\rangle$.

3.2 Argument Intersection (meet)

Given a subset T of Δ^1, we will describe the rules on it as $\{A_i >\!\!-\!\!- B_i\}_{i\in I}$. Using that representation we can consider the set $\{A_i\}_{i\in I}$ of antecedents of rules in T and the set $\{B_i\}_{i\in I}$ of consequents of those rules. If $\langle T, h\rangle$ is an argument structure for h then the set $(\mathcal{K}\cup\{B_i\}_{i\in I})^{\vdash}$ is the set of literals for which there is a subargument structure contained in $\langle T, h\rangle$.

DEFINITION 3.4 A set of rules $\{A_i >\!\!-\!\!- B_i\}_{i\in I}$ is *consistent* if and only if $\{B_i\}_{i\in I} \not\vdash \bot$.

REMARK: For arguments $\langle T, h\rangle$, T is consistent because of the non-redundancy and the $\mid\!\sim$ -consistency of arguments.

Let T be an arbitrary, but consistent, subset of Δ^1. The question is: is there a literal in $\mathcal{L}$ for which we can have an argument structure using T? The literal $(\mathcal{K}\cup\{B_i\}_{i\in I})^{\vdash}$ has that property. It also has the property of being the strongest literal, in the usual sense, with that property.

DEFINITION 3.5 Let $\langle T, h\rangle \in$ AStruc be an arbitrary argument structure, and let $\mathcal{F}$ be the family of all its subargument structures. Given $\langle T_1, h_1\rangle$, $\langle T_2, h_2\rangle$ in $\mathcal{F}$ we define the *intersection* of them as the argument structure $\langle T_3, h_3\rangle$, where

$T_3 = T_1 \cap T_2$ and h_3 is defined as $(\mathcal{K} \cup \{B_i\}_{i \in I})^{\vdash}$, where $\{B_i\}_{i \in I}$ is the set of consequents of the rules in T_3. The operation will be denoted:

$$\langle T_3, h_3 \rangle = \langle T_1, h_1 \rangle \sqcap \langle T_2, h_2 \rangle$$

PROPOSITION 3.7 *The intersection of argument structures is a well-defined operation in* $\mathcal{F}$.

PROPOSITION 3.8 *Given two argument structures* $\langle T_1, h_1 \rangle, \langle T_2, h_2 \rangle \in \mathcal{F}$, *then the intersection* $\langle T_3, h_3 \rangle = \langle T_1, h_1 \rangle \sqcap \langle T_2, h_2 \rangle$ *is such that* $\langle T_1, h_1 \rangle \succeq \langle T_3, h_3 \rangle$ *and* $\langle T_2, h_2 \rangle \succeq \langle T_3, h_3 \rangle$ *and* $\langle T_3, h_3 \rangle$ *is the maximal argument structure in* $\mathcal{F}$ *with the property of being a subargument of* $\langle T_1, h_1 \rangle$ *and* $\langle T_2, h_2 \rangle$.

PROPOSITION 3.9 (Associativity) *The intersection of arguments in* $\mathcal{F}$ *is associative, i.e. if* $\langle T_1, h_1 \rangle, \langle T_2, h_2 \rangle$, *and* $\langle T_3, h_3 \rangle$ *are subargument structures in* $\mathcal{F}$ *then*

$$(\langle T_1, h_1 \rangle \sqcap \langle T_2, h_2 \rangle) \sqcap \langle T_3, h_3 \rangle = \langle T_1, h_1 \rangle \sqcap (\langle T_2, h_2 \rangle \sqcap \langle T_3, h_3 \rangle)$$

PROPOSITION 3.10 (Commutativity) *The intersection of arguments in* $\mathcal{F}$ *is commutative, i.e. if* $\langle T_1, h_1 \rangle, \langle T_2, h_2 \rangle$ *are subargument structures in* $\mathcal{F}$ *then*

$$\langle T_1, h_1 \rangle \sqcap \langle T_2, h_2 \rangle = \langle T_2, h_2 \rangle \sqcap \langle T_1, h_1 \rangle$$

DEFINITION 3.6 Given a subfamily $\{\langle T_{i_j}, h_{i_j} \rangle\}_{i_j \in J}$ of $\{\langle T_i, h_i \rangle\}_{i \in I}$, we define the generalized intersection of the subargument structures in it as:

$$\sqcap_{j \in J} \{\langle T_{i_j}, h_{i_j} \rangle\}_{i_j \in J} = \langle \bigcap_{j \in J} T_{i_j}, (\mathcal{K} \cup \{B_i\}_{i \in I})^{\vdash} \rangle$$

where $\{B_i\}_{i\in I}$ is the set of consequents in $\bigcap_{j\in J} T_{i_j}$.

PROPOSITION 3.11 *The family $\mathcal{F}$ of subargument structures of a given argument structure $\langle T, h\rangle$ has an identity element with respect to the intersection operation. That identity element is the the argument structure $\langle T, h\rangle$.*

REMARK: Notice that at the end of the proof we have equated the literals $(\mathcal{K} \cup \{B_i\}_{i\in I})^{\vdash}$ and h_1. In fact h_1 is part of the conjunction and we can detach one of the conjuncts.

COROLLARY 3.1 *The family $\mathcal{F}$ with the intersection and combination operations defined over the argument structures forms a lattice.*

4 Justifications

In the previous chapter we introduced the notion of argument structure and defined a specificity relationship on the set of all possible argument structures. The reason to define that relationship is to be able to "select" argument structures with the characteristic of being "better" than others. In this chapter we will define the selection process.

4.1 Basic Interactions among Argument Structures

Arguments are objects that represent "pieces" of the reasoning process. They relate to each other in different ways. We have already seen an example in the *subargument* relation. Another example is the *concordance* among argument structures, *i.e.* the property which would allow to join them without producing an inconsistency. Going in the opposite direction is the *disagreement* relation that will be introduced in the next section. Some other interactions involving specificity are possible. We will introduce them now starting with those that are simplest to define.

4.1.1 Disagreement

It is possible for two argument structures to support two facts which together with the context $\mathcal{K}$ are inconsistent. We will refer to the relationship between two argument structures in that situation as *disagreement.* Next we will present the formal definition of disagreement,

DEFINITION 4.1 We say that two argument structures T_1 for h_1 and T_2 for h_2 *disagree*, denoted by

$$\langle T_1, h_1 \rangle \bowtie_{\mathcal{K}} \langle T_2, h_2 \rangle$$

if and only if $\mathcal{K} \cup \{h_1, h_2\} \vdash \bot$

The following are examples of this relationship.

EXAMPLE 4.1 $\langle \{E \succ\!\!-\!\!- \neg C\}, \neg C \rangle \bowtie_{\mathcal{K}} \langle \{A \wedge B \succ\!\!-\!\!- C\}, C \rangle$, $\mathcal{K} = \{E, A, B\}$.

EXAMPLE 4.2 $\langle \{E \succ\!\!-\!\!- \neg C\}, \neg C \rangle \bowtie_{\mathcal{K}} \langle \{A \succ\!\!-\!\!- X\}, X \rangle$, $\mathcal{K} = \{E, A, X \supset C\}$.

4.1.2 Counterargument

The counterargument relation tells us about the internal construction of an argument structure with reference to another argument structure. Is a refinement of the disagreement relation. It looks to the subarguments of a given argument structure in light of another argument, *i.e.* indicates the existence of subarguments of an argument structure which are in disagreement with the other argument. Formally:

DEFINITION 4.2 We say that an argument structure T_1 for h_1 *counterargues* an argument structure T_2 for h_2 at h, denoted by

$$\langle T_1, h_1 \rangle \overset{h}{\otimes\!\!\rightarrow} \langle T_2, h_2 \rangle$$

if and only if there exists a subargument $\langle T, h \rangle$ of $\langle T_2, h_2 \rangle$ such that $\langle T_1, h_1 \rangle \bowtie_{\mathcal{K}} \langle T, h \rangle$, *i.e.* $\langle T, h \rangle \subset \langle T_2, h_2 \rangle$ and $\mathcal{K} \cup \{h_1, h\} \vdash \bot$.

REMARK: A fact h in the conditions of the definition 4.2 will be referred to as a *counter-argument point.*

EXAMPLE 4.3 $\langle\{E \succ\!\!\!— \neg C\}, \neg C\rangle \quad \overset{C}{\otimes\!\!\rightarrow} \quad \langle\{A \wedge B \succ\!\!\!— C, C \succ\!\!\!— D\}, D\rangle$, where $\langle\{E \succ\!\!\!— \neg C\}, \neg C\rangle$ is in disagreement with the subargument $\langle\{A \wedge B \succ\!\!\!— C\}, C\rangle$ of $\langle\{A \wedge B \succ\!\!\!— C, C \succ\!\!\!— D\}, D\rangle$.

4.1.3 Defeat

The *defeat* relationship is a further refinement of counterargument, where the specificity relation comes into play. We will say that an argument structure $\langle T_1, h_1\rangle$ *defeats* another argument structure $\langle T_2, h_2\rangle$ if it is the case that $\langle T_2, h_2\rangle$ contains a subargument structure $\langle T, h\rangle$ such that $\langle T_1, h_1\rangle$ disagrees $\langle T, h\rangle$, and $\langle T_1, h_1\rangle$ is more specific than $\langle T, h\rangle$. That is:

DEFINITION 4.3 We say that an argument structure T_1 for h_1 *defeats* an argument structure T_2 for h_2, denoted by

$$\langle T_1, h_1\rangle \gg_{\text{def}} \langle T_2, h_2\rangle$$

if and only if there exists a subargument structure $\langle T, h\rangle$ of $\langle T_2, h_2\rangle$ such that:

$\langle T_1, h_1\rangle \overset{h}{\otimes\!\!\rightarrow} \langle T_2, h_2\rangle$ *i.e.* T_1 for h_1 counterargues T_2 for h_2 at h, and

$\langle T_1, h_1\rangle \succ \langle T, h\rangle$ *i.e.* T_1 for h_1 is more specific than T for h.

REMARK: A fact h in the conditions of the definition 4.3 will be referred to as a *defeater point.*

EXAMPLE 4.4 $\langle\{A \wedge B \wedge E \succ\!\!\!— \neg C\}, \neg C\rangle \quad \gg_{\text{def}} \quad \langle\{A \wedge B \succ\!\!\!— C, C \succ\!\!\!— D\}, D\rangle$, that is, the argument structure $\langle\{A \wedge B \wedge E \succ\!\!\!— \neg C\}, \neg C\rangle$ counterargues

$\langle\{A \wedge B \succ\!\!\!-\, C, C \succ\!\!\!-\, D\}, D\rangle$ at C and $\langle\{A \wedge B \wedge E \succ\!\!\!-\, \neg C\}, \neg C\rangle$ is more specific than $\langle\{A \wedge B \succ\!\!\!-\, C\}, C\rangle$.

4.2 Justifying Arguments

A fundamental issue in reasoning is to decide what the agent believes as a function of a given context and the set of defeasible rules forming his explicit knowledge. But how can he decide if a tentative conclusion is part of the implicit knowledge? Or how can he decide if that tentative conclusion is consistent with the implicit knowledge? According to our scheme this decision must be taken by analyzing what kind of support the tentative conclusion has. This can be accomplished by seeing which arguments are relevant to the conclusion.

Given a fact h, there may be several argument structures in the set $\mathbf{AStruc}(\Delta^1)$ of argument structures formed with members of Δ^1, which *support* h from the context $\mathcal{K}$. Those argument structures relate to others in $\mathbf{AStruc}(\Delta^1)$ by the defeat and counterargument relations. For an argument structure $\langle T, h\rangle$ in $\mathbf{AStruc}(\Delta^1)$, we may have a set I of argument structures which *interfere* with $\langle T, h\rangle$, *i.e.* they counterargue $\langle T, h\rangle$. In I, the set of interfering arguments, there may be some arguments which defeat $\langle T, h\rangle$. Those *defeaters* could in time be defeated. If all the defeaters are defeated, the original argument structure $\langle T, h\rangle$ becomes reinstated. The above discussion leads to an inductive definition, which is similar to Pollock's [18] and characterizes that process.

DEFINITION 4.4 Arguments are active at various levels as supporting or interfering arguments.

1. All arguments are (level 0) S-arguments (supporting arguments) and I-arguments (interfering arguments).

2. An argument $\langle T_1, h_1 \rangle$ is a (level $n+1$) S-argument if and only if there is no level n I-argument $\langle T_2, h_2 \rangle$ such that for some h, $\langle T_2, h_2 \rangle$ counterargues $\langle T_1, h_1 \rangle$ at h, *i.e.* $\nexists \langle T_2, h_2 \rangle \in$ AStruc such that, for some h, $\langle T_2, h_2 \rangle \overset{h}{\otimes\!\!\rightarrow} \langle T_1, h_1 \rangle$.

3. An argument $\langle T_1, h_1 \rangle$ is a (level $n+1$) I-argument if and only if there is no level n I-argument $\langle T_2, h_2 \rangle$ such that $\langle T_2, h_2 \rangle$ defeats $\langle T_1, h_1 \rangle$.

REMARK: A level n S-argument will be denoted by S^n-theory and a level n I-argument will be denoted by I^n-theory. Also notice that we dropped the parenthesis.

DEFINITION 4.5 We say that an argument $\langle T, h \rangle$ in AStruc *justifies* h if and only if there exists m such that for all $n \geq m$ $\langle T, h \rangle$ is an S^n-theory for h. We say that h is *justified in* $\Omega \subseteq$ AStruc if there is a $\langle T, h \rangle \in \Omega$ that justifies h.

LEMMA 4.1 *Let $\langle T, h \rangle$ be an argument structure in* AStruc, *such that $\langle T, h \rangle$ justifies h. Then every subargument $\langle R, q \rangle$ of $\langle T, h \rangle$ justifies its conclusion q.*

Proof : The proof comes from the fact that any possible defeater of $\langle R, q \rangle$ will also be a defeater for $\langle T, h \rangle$. And since $\langle T, h \rangle$ justifies h, no effective defeater exists.

We say that h is *provisionally justified* at level n iff there exists a S^n-theory which supports it. A set of provisionally justified facts is called *stable* iff every member of it is justified.

It is possible to define a sequence $\{\Sigma^n\}$ of operators over AStruc in correspondence with definition 4.4 in the following way. For a given k, let Σ^k(AStruc) be the set of h such

that there exists $\langle T, h \rangle$ that is in AStruc and is a S^k-theory; *i.e.* Σ^k produces the set of partially justified facts at level k. This definition allow us to talk about the set of justified facts in operational terms, as in the following lemmas.

LEMMA 4.2 *If* $\Sigma^n(\text{AStruc})=\Sigma^{n+1}(\text{AStruc})$ *then* $\Sigma^n(\text{AStruc})$ *is stable.*

Proof : The proof of this lemma is obvious from the definition of stable set. Once Σ has "repeated" itself, *i.e.* $\Sigma^n(\text{AStruc})= \Sigma^{n+1}(\text{AStruc})$, that means that no new interfering argument has been reinstated. Therefore, no I^n-theory can get defeated at level $n+1$ and no S^n-theory can get counterargued. □

Now the open question is whether that situation is ever reached. The next theorem will answer that question.

THEOREM 4.1 *For any Defeasible Logic Structure* $(\mathcal{K}, \Delta)$*, there is a unique stable set, and the operator* Σ *will find it.*

DEFINITION 4.6 We will refer to the stable set defined by theorem 4.1 as Σ_∞.

5 Discarding Arguments

In this section we will show some relationships among arguments and justifications aiming to find avenues pointing to efficient implementations. In that direction, it is important to find properties that will characterize arguments that can be discarded in order to reduce the size of the search space. Again, proofs are omitted.

LEMMA 5.1 *Given two argument structures* $\langle T_1, h_1\rangle$ *and* $\langle T_2, h_2\rangle$ *in* AStruc *such that* $\langle T_1, h_1\rangle \succeq \langle T_2, h_2\rangle$ *and* $\mathcal{K} \cup \{h_1\} \vdash \{h_2\}$, *then* $\langle T_1, h_2\rangle$ *is an argument structure. That is,* T_1 *is an argument for* h_2, *and* $\langle T_1, h_2\rangle \succeq \langle T_2, h_2\rangle$, *i.e.* T_1 *for* h_2 *is more specific than* T_2 *for* h_2.

LEMMA 5.2 *Given two argument structures* $\langle T_1, h_1\rangle$. *and* $\langle T_2, h_2\rangle$ *such that* $\langle T_1, h_1\rangle \succeq \langle T_2, h_2\rangle$ *and* $\mathcal{K} \cup \{h_1\} \vdash \{h_2\}$, *if* h_2 *is justified in* $\Omega \cup \{\langle T_1, h_1\rangle, \langle T_2, h_2\rangle\}$ *then* h_2 *is justified in* $\Omega \cup \{\langle T_1, h_1\rangle\}$, *where* Ω *is any subset of* AStruc.

LEMMA 5.3 *Given two argument structures* $\langle T_1, h_1\rangle$ *and* $\langle T_2, h_2\rangle$ *such that* $\langle T_1, h_1\rangle$ covers $\langle T_2, h_2\rangle$, *i.e.* $\langle T_1, h_1\rangle \succeq \langle T_2, h_2\rangle$ *and* $\mathcal{K} \cup T_1 \mid\!\sim x$, *for all* x *in* $\mathrm{Co}(T_2)$, *then if* p *is justified in* $\Omega \cup \{\langle T_1, h_1\rangle, \langle T_2, h_2\rangle\}$, p *is also justified in* $\Omega \cup \{\langle T_1, h_1\rangle\}$, *where* Ω *is any subset of* AStruc.

PROPOSITION 5.1 *Given* $\langle T_1, h_1\rangle$, $\langle T_2, h_2\rangle$, *and* $\langle T, h\rangle$ *in* AStruc, *where* $\langle T_1, h_1\rangle$ *covers* $\langle T_2, h_2\rangle$, *then if* $\langle T_2, h_2\rangle$ *contains a subargument structure* $\langle R, p\rangle$ *such that* $\langle R, p\rangle \otimes\!\!\stackrel{p}{\rightarrow} \langle T, h\rangle$, *then* $\langle T_1, h_1\rangle$ *contains a subargument structure* $\langle S, p\rangle$ *such that* $\langle S, p\rangle \otimes\!\!\stackrel{p}{\rightarrow} \langle T, h\rangle$.

PROPOSITION 5.2 *Given* $\langle T_1, h_1\rangle$, $\langle T_2, h_2\rangle$, *and* $\langle T, h\rangle$ *in* AStruc *such that* $\langle T_1, h_1\rangle$ *covers* $\langle T_2, h_2\rangle$, *then if* $\langle T_2, h_2\rangle \gg_{\mathrm{def}} \langle T, h\rangle$ *then* $\langle T_1, h_1\rangle \gg_{\mathrm{def}} \langle T, h\rangle$.

Given $\Omega \subseteq$ AStruc and $\langle T_1, h_1\rangle$, $\langle T_2, h_2\rangle$, $\langle T, h\rangle \in$ AStruc, where $\langle T_1, h_1\rangle$ covers $\langle T_2, h_2\rangle$, define

$$\Omega_{\mathrm{big}} = \Omega \cup \{\langle T, h\rangle, \langle T_1, h_1\rangle, \langle T_2, h_2\rangle\}$$

$$\Omega_{\text{small}} = \Omega \cup \{\langle T, h\rangle, \langle T_1, h_1\rangle\}$$

PROPOSITION 5.3 *If $\langle T, h\rangle$ is a S^n-theory in Ω_{big} then $\langle T, h\rangle$ is a S^n-theory in Ω_{small}.*

6 Some Interesting Examples

We will show some examples presented in the literature of defeasible reasoning to show the behavior of the system.

EXAMPLE 6.1 [Opus does not fly]

An example of how information regarding a subclass overrides more general information corresponding to superclasses.

Birds tend to fly	$(B(x)\succ\!\!-F(x))$
Penguins tend to not fly	$(P(x)\succ\!\!-\neg F(x))$
All penguins are birds	$(P(x) \supset B(x))$
Opus is a Penguin	$(P(opus))$
Does Opus fly?	$(F(opus)?)$

The context and defeasible rule set are $\mathcal{K} = \{P(opus), P(opus) \supset B(opus)\}$, and $\Delta = \{B(opus)\succ\!\!-F(opus), P(opus)\succ\!\!-\neg F(opus)\}$.

Two argument structures are interesting:

$$\langle T_1, F(opus)\rangle = \langle\{B(opus)\succ\!\!-F(opus)\}, F(opus)\rangle$$

$$\langle T_2, \neg F(opus)\rangle = \langle\{P(opus)\succ\!\!-\neg F(opus)\}, \neg F(opus)\rangle$$

We have the following disagreement $\langle T_1, F(opus)\rangle \bowtie_{\mathcal{K}} \langle T_2, \neg F(opus)\rangle$.

Moreover $\langle T_2, \neg F(opus)\rangle \overset{\neg F(opus)}{\otimes\!\!\rightarrow} \langle T_1, F(opus)\rangle$.

But $\langle T_2, \neg F(opus)\rangle \succ \langle T_1, F(opus)\rangle$.

Therefore, $\langle T_2, \neg F(opus)\rangle \gg_{\text{def}} \langle T_1, F(opus)\rangle$, hence $\langle T_2, \neg F(opus)\rangle$ justifies $\neg F(opus)$.

EXAMPLE 6.2 [Nixon Diamond]

This canonical example is devised to show how the reasoner behaves in ambiguous situations and is due to Reiter [21].

Quakers tend to be pacifist	$(Q(x) \succ\!\!- P(x))$
Republicans tend to be non-pacifist	$(R(x) \succ\!\!- \neg P(x))$
Nixon is a quaker	$(Q(nix))$
Nixon is a republican	$(R(nix))$
Is Nixon pacifist?	$(P(nix)?)$

Clearly, there are three possible behaviors. The first, which is clearly undesirable, represents the behavior of simple shortest path reasoner and will give one of the two possibilities arbitrarily or give both answers as true. The second, which is the behavior of reasoners using the inferential distance ordering instead of length of the path [26], will give two answers, leaving the decision to whoever uses the system. This kind of reasoner is called *credulous* because it gives good standing to all the possible conclusions. The last, the so called *skeptical* reasoner, does not decide about ambiguity [8] by not giving any answer. Our reasoner is skeptical.

The context and defeasible rule set are $\mathcal{K} = \{R(nix), Q(nix)\}$, and $\Delta = \{Q(nix) \succ\!\!- P(nix), R(nix) \succ\!\!- \neg P(nix)\}$ respectively. We have two argument structures, one for $P(nix)$ and one for $\neg P(nix)$, namely,

$$\langle T_1, P(nix)\rangle = \langle \{Q(nix) \succ\!\!- P(nix)\}, P(nix)\rangle$$

$$\langle T_2, \neg P(nix)\rangle = \langle \{R(nix) \succ\!\!- \neg P(nix)\}, \neg P(nix)\rangle$$

None of those argument structures defeats the other; they interfere and our reasoner does not give an answer.

EXAMPLE 6.3 [Cascaded Ambiguities]

This example is an extension of the Nixon Diamond constructed to show how simple-minded skeptical reasoners can be fooled to believe in the militarism (non-anti-militarism) of Nixon [8].

Quakers tend to be pacifist	$(Q(x) \succ\!\!- P(x))$
Republicans tend to be non-pacifist	$(R(x) \succ\!\!- \neg P(x))$
Pacifist tend to be anti-military	$(P(x) \succ\!\!- A(x))$
Republicans tend to be football fans	$(R(x) \succ\!\!- F(x))$
Football fans tend to be non-anti-military	$(F(x) \succ\!\!- \neg A(x))$
Nixon is a quaker	$(Q(nix))$
Nixon is a republican	$(R(nix))$
Is Nixon anti-military?	$(A(nix)?)$

The context and defeasible rule set are $\mathcal{K} = \{R(nix), Q(nix)\}$, and

$\Delta = \{Q(nix) \succ\!\!- P(nix),\ R(nix) \succ\!\!- \neg P(nix),\ P(nix) \succ\!\!- A(nix),$

$R(nix) \succ\!\!- F(nix),\ F(nix) \succ\!\!- \neg A(nix)\}$.

We have two argument structures, one for $A(nix)$ and one for $\neg A(nix)$, namely,

$$\langle T_1, A(nix)\rangle = \langle \{Q(nix) \succ\!\!- P(nix), P(nix) \succ\!\!- A(nix)\}, A(nix)\rangle$$

$$\langle T_2, \neg A(nix)\rangle = \langle \{R(nix) \succ\!\!- F(nix), F(nix) \succ\!\!- \neg A(nix), \}, \neg A(nix)\rangle$$

None of those argument structures defeat the other and our reasoner remains skeptical.

Notice that some skeptical reasoners will consider the "path" (using inheritance reasoners terminology), $\{Q(nix) \succ\!\!- P(nix), P(nix) \succ\!\!- A(nix)\}$ as being preempted by $\{R(nix) \succ\!\!- \neg P(nix)\}$ and hence leaving $\{R(nix) \succ\!\!- F(nix), F(nix) \succ\!\!- \neg A(nix)\}$ free to support the conclusion about Nixon being non-anti-military. That situation does not arise in our case. The argument structure $\langle \{R(nix) \succ\!\!- \neg P(nix)\}, \neg P(nix) \rangle$ counterargues $\langle T_1, A(nix) \rangle$ at $P(nix)$, but $\langle \{R(nix) \succ\!\!- \neg P(nix)\}, \neg P(nix) \rangle$ is not more specific than $\langle T_1, A(nix) \rangle$. Therefore, $\langle \{R(nix) \succ\!\!- \neg P(nix)\}, \neg P(nix) \rangle$ does not defeat $\langle T_1, A(nix) \rangle$.

EXAMPLE 6.4 [Royal African Elephants]

This example deals with "on-path vs off-path preemption" and is due to Sandewall [22], in the context of inheritance reasoners.

Elephants tend to be gray	$(E(x) \succ\!\!- G(x))$
Royal elephants tend to be non-gray	$(R(x) \succ\!\!- \neg G(x))$
Royal elephants are elephants	$(R(x) \supset E(x))$
African elephants are elephants	$(A(x) \supset E(x))$
Clyde is a royal elephant	$(R(clyde))$
Clyde is an african elephant	$(A(clyde))$
Is Clyde non-gray?	$(\neg G(clyde))$

The context and defeasible rules are $\mathcal{K}$= $\{R(clyde), A(clyde),\ R(clyde) \supset E(clyde),\ A(clyde) \supset E(clyde)\}$, and Δ= $\{E(clyde) \succ\!\!- G(clyde),\ R(clyde) \succ\!\!- \neg G(clyde)\}$ respectively.

We have three argument structures, two for $G(clyde)$ and one for $\neg G(clyde)$, namely,

$$\langle T_1, G(clyde)\rangle = \langle \{E(clyde) \succ\!\!- G(clyde)\}, G(clyde)\rangle$$

$$\langle T_2, G(clyde)\rangle = \langle \{E(clyde) \succ\!\!- G(clyde)\}, G(clyde)\rangle$$

$$\langle T_3, \neg G(clyde)\rangle = \langle \{R(clyde) \succ\!\!- \neg G(clyde)\}, \neg G(clyde)\rangle$$

Clearly, the more specific argument structure is $\langle T_3, \neg G(clyde)\rangle$, matching our intuitions.

7 A Justification Finder

Implementations of defeasible reasoners are rarely seen beasts. An early attempt to introduce defeasible reasoning programming with specificity was Nute's d-Prolog [17, 16]. The language of d-Prolog provides facilities to define *absolute rules*, "every bat is a mammal", *defeasible rules*, "birds fly", and *defeater rules*, "sick birds do not fly". The purpose of defeater rules was to account for the exceptions to defeasible rules. For instance, given the defeasible rule "birds fly", the defeater rule "sick birds do not fly" will stop us from concluding that "Tweety flies", in the presence of the fact "Tweety is a sick bird".

The system jf is a practical implementation of the theoretical ideas introduced in previous chapters. It was used to explore and validate those ideas and it has produced valuable feedback in understanding the computational aspects of performing defeasible reasoning.

jf provides support for defeasible reasoning in a Prolog environment, like Nute's d-Prolog. The complete Prolog language is available, and only a few new predicates are introduced, extending the reserved words of the language. The choice of Prolog as the implementation language was made for two reasons. First, Prolog is a logic programming language and therefore is close in spirit to our defeasible reasoner. Second, meta-programming is a natural activity in Prolog given the equivalence of programs and data, making the writing of an experimental interpreter much easier.

We claim that adding special kinds of defeasible rules amounts to a retreat from the goal of having a declarative language. It is true that reaching that goal is a difficult task, but the implementation of experimental systems, such as jf, should strive to maintain that ideal rather than start by giving it up. This constraint in the design of a language translates

the burden of computing to the process of searching the solution space because there is no "procedural" guidance. That search space in turn can be quite large for non trivial situations. Much of the effort expended in the implementation was put to this task of doing an efficient search.

Logic programming has a clear semantics based on the interpretation of programs as Horn clauses (see [11] and next section below). That characteristic cannot be maintained for logic programming languages such as Prolog due to the introduction of extra-logical features such as cut. Nevertheless, we will keep that clear semantics in our language by not adding any extra-logical devices. That semantics is a reduction of the semantics introduced in the previous chapter, from full first order logic to Horn clauses. We will assume that the reader has familiarity with Logic Programming concepts. For an introduction to the theoretical foundations of logic programming see Lloyd [9]. We will introduce some definitions in the next section in order to have agreement over the terminology.

7.1 The Language Implemented

The basic ideas of logic programming are introduced here using the standard notation for them. We will slightly modify that notation as we introduce our language.

DEFINITION 7.1 A *definite clause* is a clause of the form:

$$B \Leftarrow A_1, \ldots, A_n$$

with only one atom as the consequent. The consequent B is called the *head* and the antecedent $A_1, \ldots, A_n$ is called the *body* of the definite clause.

It is customary to regard all clauses as implications, even though they have no head or body. We will alter this for our language in a way that is consistent with this presentation. The reasons for that modification will be given below.

DEFINITION 7.2 A *definite goal* is a clause of the form: $\Longleftarrow A_1, \ldots, A_n$ *i.e.* a definite clause with an empty consequent. The A_i are sometimes called *subgoals* of the goal.
A *unit clause* is a clause of the form: $B \Longleftarrow$, *i.e.* a definite clause with an empty body. We will alter this representation introducing the special atom *true*. Our unit clauses will be written:

$$B \Longleftarrow true$$

Unit clauses are also called *facts*.

DEFINITION 7.3 A *Horn clause* is clause which is either a definite clause or a definite goal.

We have extended the representation in two ways. First, we added defeasible clauses, and second, we introduced a relation *neg* used to represent negative facts.

DEFINITION 7.4 A *defeasible clause* is a clause of the form:

$$B \prec\!\!\!- A_1, \ldots, A_n$$

with only one atom as the consequent. The consequent B is called the *head* of the defeasible rule and the antecedent $A_1, \ldots, A_n$ is called the *body* of the defeasible clause.

The *neg* relation will allow the representation of negative facts in the system. Negation is handled in the same way as proposed by Nute [17, 16]. This relation is not related in any way to *negation as failure* and its only meaning is to refer to a negative fact. Negative facts

relate to positive facts in the usual way. The system will treat the relation *neg* as a prefix forming part of the "name" of the atom and not as an operator. The system will recognize that *neg neg* $A = A$. That is, the goal *neg* A will be assumed as a consequence of a set R of definite and defeasible clauses if and only if *neg* A is deducible from R via a finite number of applications of modus ponens. The goal *neg neg* A will be assumed as a consequence of a set R of definite and defeasible clauses if and only if *neg neg* A, or A, is deducible from R via a finite number of applications of modus ponens. Thus, the relation *neg* does not have any special status; the system will treat the atom *neg* A in the same way as any other atom C.

This form of negation is in no way related to "negation as failure" [1] as usually defined in Prolog. Negation in Prolog, the prefix operator *not*, is defined by saying that the goal *not* A succeeds if and only if the goal A is "finitely failed". In that way, the operator *not* is only a partial form of negation in first order logic. Several other forms of negation, with increasing power, can be implemented (see Naish [13]), but it would require to change the resolution method. We would like to keep Prolog as our base language and take advantage of the efficient compilers available.

Our *neg* operator can appear in the head of the rules, defeasible and otherwise. For instance,

$$neg\ A \Longleftarrow true$$

$$neg\ A \Longleftarrow neg\ B,\ C$$

$$neg\ A \prec\!\!\!- B,\ C,\ neg\ D$$

are legal rules. Notice that the first rule is asserting a negative fact.

DEFINITION 7.5 A *knowledge base* $\mathbb{K}$ is a finite set of definite clauses and defeasible clauses, possibly containing atoms affected by the *neg* relation. A knowledge base is the equivalent of what previously was called a Defeasible Logic Structure. In a knowledge base $\mathbb{K}$ the set $\mathcal{K}$ will be represented using definite clauses, and the set Δ will be represented using defeasible clauses.

7.2 Finding Justifications

The interpreter will work following the lines of the proof of theorem 4.1 taking advantage of the inference mechanism of Prolog.

The input to jf is a knowledge base $\mathbb{K}$, and a ground query Q. The contents of the knowledge base were described in the previous section. A ground query Q is a ground instance of an atom, possibly affected with the prefix *neg*. The justifier is invoked by issuing the command:

analyze(Q)

which will start the process of testing whether there is an undefeated argument which supports Q from the contents of $\mathbb{K}$.

If the search finds a justification the output of the system for such a query will be one of the argument structures that are justifying Q, and all the possible defeaters that were considered. All the justifiers can be obtained by rejecting the answer, and forcing the system to keep searching.

If the answer is negative, the system will have two possible answers. The query Q has no supporting argument. Or even though arguments can be constructed to support it, all of them were defeated. In the latter case, the system will return all the potential justifiers, already defeated, with its associated defeaters. We will disregard the uninteresting case when Q has no supporting argument.

The process begins by attempting to construct an argument for the given query Q. Arguments for Q are constructed by using backward chaining over the knowledge base. We will follow Shapiro's [25] terminology. A *ground reduction* of a goal G in a knowledge base IK is the replacement of G by the body of a ground instance of a clause (definite or defeasible), whose head is identical to G. A *defeasible inference tree* consists of nodes and edges which represent the goals reduced during the construction. The root of the tree is the original query and the nodes are the goals reduced during the backward chaining. Edges represent the relation between the head of the rule used in the reduction and the atoms in the body of that rule. The backward chaining on a node G stops whenever G is supported by an unit clause, *i.e.* a clause like $G \Longleftarrow true$. The following example will help to describe the process:

EXAMPLE 7.1 Assume the following knowledge base IK:

$flies(x) \prec bird(x)$	usually birds fly
$neg\ flies(x) \prec penguin(x)$	usually penguins do not fly
$bird(x) \Longleftarrow penguin(x)$	penguins are birds
$penguin(opus) \Longleftarrow true$	opus is a penguin

After the query "*analyze*(*flies*(opus))", the system will form the argument { *flies*(opus) $-\!\prec$ *bird*(opus), *bird*(opus) $\Longleftarrow$ *penguin*(opus), *penguin*(opus) $\Longleftarrow$ *true*} by backward chaining from *flies*(opus).

It is easy to see that the system will always form the most specific argument. If the system is forced to backtrack from an unit clause $G \Longleftarrow$*true*, it will not attempt to find support for G in other clauses. Following those clauses will only produce a less specific argument. This observation was already suggested in the proof of theorem 4.1.

After forming an argument, the system will try to find counterarguments for the recently formed argument by backward chaining from the negation of atoms in the original argument. Actually, the system will form a set with the atoms in the argument, and will add to that set any atom that is derivable from those atoms and the definite clauses in $\mathbb{K}$. For instance, in the example above, it will find the counterargument { *neg flies*(opus) $-\!\prec$ *penguin*(opus), *penguin*(opus) $\Longleftarrow$ *true* }.

Finally, the system will test the argument and the counterargument for specificity using the activation models of the argument (see Simari [24]) and the activation models of the counterargument. In short, M is an activation model for $\langle T, h\rangle$ if M is a model of $\mathcal{K}_{\mathcal{N}}$ and M is also a model for some $e \in Sent_{\mathcal{C}}(\mathcal{L})$ and for the rules that form the subset T' of T such that $\mathcal{K}_{\mathcal{N}} \cup \{e\} \cup T' \mid\!\sim h$. Using those criteria in our example, we will find that any activation model for the argument for *neg flies*(opus) is an activation model for the argument for *flies*(opus). But there is an activation model for *flies*(opus) which is not an activation model for *neg flies*(opus), namely the one where *bird*(opus) is true but *penguin*(opus) is not.

If the argument is defeated, as is the case in the example, the system will backtrack in

the process that formed the original argument, discarding the last rule added to the tree, trying to replace it with another. If it finds one, the process of finding and testing defeaters is repeated. Otherwise, further backtracking is necessary. This process will continue until an undefeated argument is produced or all the backtracking possibilities are exhausted.

The following is a simple abstract interpreter written in Prolog, its parameters are explained in the comments. Where *Goal* is a ground atom for which a justification is sought, *Argument* is an output parameter which will contain an argument for *Goal* if there is one.

```
% analyze(+Goal, -Argument, -Defeater_List)
% Input:  Goal is ground atom for which a justification is sought.
% Output: Argument will contain an argument for Goal;
%         Argument can be empty if no argument exists.
%         Defeater_List will contain the defeaters of Argument;
%         Defeater_List will empty if no defeater exists.

analyze(Goal, Argument, Defeater_List) :-
        construct_argument(Goal, Argument),
        find_defeaters(Argument, Defeater_List),
        test_defeaters(Argument, Defeater_List).
```

Finally, we would like to remark how much the implementation of this system has helped us to understand some of the complex issues involved in performing defeasible reasoning with specificity. Meanwhile, solving many of the theoretical questions has helped the im-

plementation. Theorem 4.1 is a good example of that. The statement of theorem 4.1 gives a termination condition for the level n computation, but certain objects constructed during the proof, *e.g.* the argumental lines, showed the way to the efficient implementation of the system. Even though the definition of level n computation is clear and concise, it does not yield an easy implementation in logic programming. On the other hand, argumental lines are especially suited for that purpose. They are also very important in the consideration of partial computation. This topic will be addressed elsewhere.

8 Conclusions

In this paper we have presented a mathematical approach to defeasible reasoning. This approach is based on the notion of specificity introduced by Poole and the general theory of warrant as presented by Pollock. Poole's approach to specificity was correct but he stopped short of presenting a complete approach to it. We proved that an order relation can be introduced among equivalence classes under the equi-specificity relation. Poole did not pursue operational aspects of applying specificity. We did that here.

Pollock has suggested an operational framework for performing reasoning, but he dismissed useful and prevalent generalizations of specificity. Taking his definition of warrant, we have applied it and transformed it into a justification schema which defines the set of justified facts from a given Defeasible Logic Structure. One result of this paper is a theorem that ensures the termination of the process of finding the justified facts. The proof of that result is based on the order relation mentioned above.

In order to implement the theoretical ideas, a suitable restriction of the language has been defined. The language used to represent the context $\mathcal{K}$ has been restricted to a subset of first order logic, Horn clauses, and the language used to represent defeasible rules in Δ has been restricted in a similar way, to a Horn-clause-like syntax. The interpreter was written in Prolog, and running on top of it provides a defeasible reasoning tool for Prolog.

The implementation of the system has taken advantage of the theoretical findings. The general mechanism used in the implementation to find justifications is based on the structures built in proof of the theorem on termination. The process used to compare two

argument structures for specificity is based on semantical work that is not reported here.[4] Two more lemmas, 2.4 and 2.5, define a reduced search space. Meanwhile, it is the prospect of implementation that suggested many of these theorems.

In comparison with inheritance, this system generalizes the idea of path, clarifies the logic of reinstatement, and even in its Horn clause form, provides more expressive language. In comparison with [10], this system shares the same spirit and many of its syntactic considerations, though reproduces almost none of the details. In particular, $\mathcal{K}$, Δ, and $\succ$—are taken from Loui, which in turn originate with Kyburg. Further, Loui's definition of arguments as digraphs confuses definitional and implementational issues, which this paper separates. In comparison with Geffner, this approach represents an alternative, older paradigm, based on arguments instead of irrelevance.

To summarize, the introduction of Defeasible Logic Structures as a way of performing defeasible reasoning represents the unification of ideas in a formal and concise system which exhibits a correct, and uniform, behavior when applied to the benchmark examples in the literature. The investigation of the theoretical issues has aided the study of how this kind of reasoner can be realized on a computer, leading to an efficient implementation. We believe that the presentation here may have more permanence than past approaches to defeasible argument.

[4] but is available in the dissertation [23].

9 Acknowledgment

This research has been conducted within the Washington University Center for Intelligent Computer Systems. An industrial consortium supported by McDonnell Douglas Corporation and Southwestern Bell Corporation.

References

[1] K. L. Clark. Negation as Failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[2] J. P. Delgrande. An Approach to Default Reasoning Based on a First-Order Conditional Logic. In *Proceedings of the National Conference on Artificial Intelligence*, pages 340–345. AAAI, 1987.

[3] H. B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.

[4] G.Birkhoff and T.C.Bartee. *Modern Applied Algebra.* McGraw-Hill, 1970.

[5] H. Geffner and J. Pearl. A Framework for Reasoning with Defaults. Technical Report TR-94III CSD-870058, Cognitive Systems Lab., UCLA, Sept 1989. To appear in *Defeasible Reasoning and Knowledge Representation*, Kluwer Pub, 1989.

[6] C. Glymour and R. H. Thomason. Default Reasoning and the Logic of Theory Perturbation. In *Nonmonotonic Reasoning Workshop*, pages 93–102, Menlo Park, CA, 1984. American Association for Artificial Intelligence.

[7] J. Henry E. Kyburg. *Logical Foundations of Statistical Inference.* Reidel, 1974.

[8] J. F. Horty, R. H. Thomason, and D. S. Touretzky. A Skeptical Theory of Inheritance in Nonmonotonic Semantic Networks. Technical Report CMU-CS-87-175, Carnegie Mellon University, Computer Science Department, Oct 1987.

[9] J. Lloyd. *Fundations of Logic Programming.* Springer-Verlag, New York, second edition edition, 1987.

[10] R. P. Loui. Defeat Among Arguments: A System of Defeasible Inference. *Computational Intelligence*, 3(3), 1987.

[11] A. Martelli and G. Rossi. On the Semantics of Logic Programming Languages. In E. Shapiro, editor, *Third International Conference on Logic Programming*, pages 327–334. Springer-Verlag, July 1986.

[12] E. Mendelson. *Introduction to Mathematical Logic.* Wadsworth & Brooks/Cole, third edition, 1987.

[13] L. Naish. *Negation and Control in Prolog (LNCS-238).* Springer-Verlag, 1986.

[14] E. Neufeld and D. Poole. Probabilistic Semantics and Defaults. In R. Schachter, T. Levitt, J. Lemmer, and L. Kanal, editors, *Uncertainty in AI.* North-Holland, 1989 (in press).

[15] D. Nute. A Non-monotonic Logic Based on Conditional Logic. Technical Report ACMC Research Rep. 01-0007, University of Georgia, Athens, GA, 1985.

[16] D. Nute. Defeasible Reasoning. In J. H. Fetzer, editor, *Aspects of Artificial Intelligence*, pages 251–288. Kluwer Academic Publishers, Norwell, MA, 1988.

[17] D. Nute and M. Lewis. d-Prolog: A Users Manual. Technical Report ACMC 01-0017, University of Georgia, Athens, 1986.

[18] J. L. Pollock. Defeasible Reasoning. *Cognitive Science*, 11:481–518, 1987.

[19] D. Poole. A Logical Framework for Default Reasoning. *Artificial Intelligence*, 36(1):27–47, 1988.

[20] D. L. Poole. On the Comparison of Theories: Preferring the Most Specific Explanation. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 144–147. IJCAI, 1985.

[21] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13(1,2):81–132, Apr 1980.

[22] E. Sandewall. Non-monotonic Inference Rules for Multiple Inheritance with Exceptions. *Proceedings of the IEEE*, 74:481–518, 1986.

[23] G. R. Simari. *A Mathematical Treatment of Defeasible Reasoning and its Implementation.* PhD thesis, Department of Computer Science, Washington University, St. Louis, MO, December 1989.

[24] G. R. Simari. On the Logic of Defeasible Reasoning. Technical Report WUCS-89-12, Washington University, Department of Computer Science, April 1989.

[25] L. Sterling and E. Shapiro. *The Art of Prolog.* The MIT Press, Cambridge, Massachusetts, 1986.

[26] D. S. Touretzky. *The Mathematics of Inheritance Systems.* Morgan Kaufmann Publishers, Los Altos, CA, 1986.