Computing Generalized Specificity

Frieder Stolzenburg, stolzen@uni-koblenz.de*
Alejandro J. García, ajg@cs.uns.edu.ar**
Carlos I. Chesñevar, cic@cs.uns.edu.ar**
Guillermo R. Simari, grs@cs.uns.edu.ar**

* *Univ. Koblenz-Landau, Universitätsstr. 1, 56070 Koblenz, GERMANY*

** *Univ. Nacional del Sur, Av. Alem 1253, (B8000CPB) Bahía Blanca, ARGENTINA*

ABSTRACT. *Most formalisms for representing common-sense knowledge allow incomplete and potentially inconsistent information. When strong negation is also allowed, contradictory conclusions can arise. A criterion for deciding between them is needed. The aim of this paper is to investigate an inherent and autonomous comparison criterion, based on specificity as defined in [POO 85, SIM 92]. In contrast to other approaches, we consider not only defeasible, but also strict knowledge. Our criterion is context-sensitive, i. e., preference among defeasible rules is determined dynamically during the dialectical analysis.*
We show how specificity can be defined in terms of two different approaches: activation sets *and* derivation trees. *This allows us to get a syntactic criterion that can be implemented in a computationally attractive way. The resulting definitions may be applied in general rule-based formalisms. We present theorems linking both characterizations. Finally we discuss other frameworks for defeasible reasoning in which preference handling is considered explicitly.*

RÉSUMÉ. *A défi nir par la commande* \resume{...}

KEYWORDS: *defeasible reasoning; knowledge representation; logic programming; nonmonotonic reasoning.*

MOTS-CLÉS : *A défi nir par la commande* \motscles{...}

1. Introduction

1.1. *Background*

Formalisms for representing common-sense knowledge usually handle incomplete and potentially inconsistent information. In such formalisms, contradictory conclusions can arise, which prompts for a criterion for deciding between them. Several extensions of logic programming (LP), default reasoning systems, defeasible

logics, and defeasible argumentation formalisms consider priorities over competing rules [ANT 00, COV 88, DIM 95, GEL 97, KAK 94, WAN 97], in order to decide between contradictory conclusions. However, these priorities must be supplied by the programmer, establishing explicitly relations between rules.

Another problem, pointed out by Dung and Son in [DUN 96], is that several formalisms "enforce" the principle of reasoning with specificity by first determining a set of priority orders between default rules of a set D, using the information given by a domain knowledge K. The problem is that the resulting semantics is rather weak, in the sense that priorities are defined independently of the set E of evidence. Therefore, if the set E changes, the previous fixed priorities could not behave as expected. On the contrary, this evidence-sensitivity can be naturally captured in argumentation-theoretic approaches as shown in [DUN 96, GAR 98, SIM 92] and also here.

In [DUN 96], a transformation from the proposed underlying language into extended logic programming [GEL 90a] is given. However, this transformation encodes the specificity criterion with program rules, forcing re-encoding in the presence of changes in the program. In our approach specificity will be inferred directly from the program rules without any intermediate step. Our approach also takes into consideration the background knowledge B that was assumed empty in [DUN 96]. Dealing with background knowledge (i. e. adding strict rules) is not a trivial matter, because then we have to take into account also the conclusions that are implied by this background knowledge, which has to be considered in the dialectical process when comparing arguments (see also Section 3.2).

1.2. *Motivation*

The aim of this paper is to investigate beyond explicit comparison between rules, looking forward for a more autonomous comparison criterion, based on specificity as defined in [POO 85, SIM 92]. In contrast to other approaches, we consider not only defeasible, but also strict knowledge. In our setting, arguments will be basically *defeasible proofs* involving both defeasible and strict knowledge, which may support contradictory conclusions, so that a comparison criterion is needed to decide between them. Our criterion for comparing arguments, namely *specifi city*, is context-sensitive. This means that preference among defeasible rules is determined dynamically during the dialectical analysis (see also the examples in Section 4.1).

We show how this criterion can be redefined in terms of two different approaches: *activation sets* and *derivation trees*. This allows us to get a syntactic criterion that can be implemented in a computationally attractive way. The resulting definitions may be applied in arbitrary generic rule-based formalisms. As a basis of our presentation we will use *Defeasible Logic Programming* (DeLP) [GAR 97, GAR 98], where a comparison for arguments based on specificity is given. In DeLP (as in many defeasible logics and defeasible argumentation formalisms), there is a distinction between strict

rules and defeasible rules. Specificity in DeLP takes into consideration both kinds of rules.

Originally, this research has been motivated by the programming of autonomous agents for the RoboCup [MUR 01]. Since agents must be able to cope with contradictory knowledge, defeasible reasoning should be employed for agent programming. Defeasible logic programming is able to extend the logic-based approach for multiagent systems as presented in [MUR 01].

This paper is organized as follows. First, in Section 2 we introduce the fundamentals of DeLP. In Section 3, a definition of generalized specificity will be given, and two computationally attractive ways of comparing arguments by means of specificity in a logic programming framework will be developed. Finally, in Section 4, we discuss other frameworks for defeasible reasoning in which preference handling is considered explicitly, contrasting them with our approach. We will end with concluding remarks in Section 5.

2. Defeasible Logic Programming

2.1. *Defeasible Programs*

The DeLP language [GAR 97, GAR 98] is defined in terms of two disjoint sets of rules: a set of *strict rules* for representing strict (sound) knowledge, and a set of *defeasible rules* for representing tentative information. Rules will be defined using *literals*. A literal L is an atom p or a negated atom $\sim p$, where the symbol $\sim$ represents *strong negation*. We define this formally as follows:

Definition 2.1 (Strict rule) *A strict rule is an ordered pair, conveniently denoted $Head \leftarrow Body$, whose first member, $Head$, is a literal, and whose second member, $Body$, is a finite set of literals. A strict rule with the head L_0 and body $\{L_1, \ldots, L_n\}$ can also be written as $L_0 \leftarrow L_1, \ldots, L_n$. As usual, if the body is empty, then a strict rule becomes $L \leftarrow true$ (or simply L) and it is called a* fact.

The syntax of strict rules corresponds to *basic rules* in logic programming [LIF 96], but we call them "strict" in order to emphasize the difference to the "defeasible" ones (see below). There is no contraposition for rules, i. e., $a \leftarrow b$ is not equivalent to $\sim b \leftarrow \sim a$. Defeasible rules add a new representational capability for expressing a weaker link between the head and the body in a rule [SIM 92].

Definition 2.2 (Defeasible rule) *A defeasible rule is an ordered pair, conveniently denoted $Head \prec Body$, whose first member, $Head$, is a literal, and whose second member, $Body$, is a finite and non-empty set of literals. A defeasible rule with head L_0 and body $\{L_1, \ldots, L_n\}$ can also be written as $L_0 \prec L_1, \ldots, L_n$ where $n \geq 1$.*

A defeasible rule with an empty body (i. e. $n = 0$ in this case) is called a *presumption* [GAR 97, GAR 00]. Technically, it is possible to introduce presumptions into a framework for defeasible argumentation with specificity. However, this might lead to counterintuitive results when comparing arguments by the definition of specificity we adopt in this paper (see Definition 3.1). For instance, two arguments solely based on presumptions are not comparable wrt. specificity according to Definition 3.1, although they should, if the set of presumptions used in one argument are a proper subset of the set of presumptions used in the other argument. Therefore we will exclude presumptions from our object language. For an in-depth analysis of presumptions with respect to DeLP the reader is referred to [GAR 00].

Syntactically, the symbol "$-\!\prec$" is all that distinguishes a defeasible rule from a strict one. Pragmatically, a defeasible rule is used to represent defeasible knowledge, i. e. tentative information that may be used if nothing could be posed against it.

Definition 2.3 (Defeasible logic program) *A* defeasible logic program (DLP) *is a finite set of strict and defeasible rules where literals may have variable or constant parameters. We do not consider the case with general functions here. If $\mathcal{P}$ is a* DLP*, we will distinguish the subset Π of strict rules in $\mathcal{P}$, and the subset Δ of defeasible rules in $\mathcal{P}$. When required, we will denote $\mathcal{P}$ as* (Π, Δ).

Example 2.4 *The following is a* DLP *where strict and defeasible rules have been separated for the convenience of presentation. It models a fragment of the soccer domain. In the following, predicate and constant symbols (e. g. $eager$ and $diego$, respectively) begin with lower-case letters, while variable symbols (e. g. X) begin with capital letters as in Prolog programs.*

Δ	Π
$kick(X) -\!\prec player(X)$	$player(X) \leftarrow libero(X)$
$\sim kick(X) -\!\prec libero(X)$	$player(X) \leftarrow goalie(X)$
$kick(X) -\!\prec libero(X), eager(X)$	$\sim kick(X) \leftarrow goalie(X)$
	$eager(diego)$
	$libero(diego)$
	$goalie(oli)$

Nute's defeasible logic [COV 88, NUT 94], recent extensions of defeasible logic [ANT 00, MAH 98] and some defeasible argumentation formalisms [HOR 94, PRA 97, VRE 97] also make use of defeasible and strict rules for representing knowledge. However, in most of these formalisms a priority relation among rules must be explicitly given with the program in order to handle contradictory information. In DeLP, an argumentation formalism for deciding between contradictory goals is used.

2.2. *Defeasible Derivations*

A *defeasible derivation* for a literal h from a DLP $\mathcal{P}$ is a finite set of strict and defeasible rules, obtained like an SLD-derivation, as defined in [LLO 87], but considering the negation symbol "$\sim$" as part of the predicate name and not taking into consideration the type of the rule. Since we do not need the notion of SLD-derivation explicitly in this context, we refer the reader to [LLO 87] for more details. Nevertheless, an SLD-derivation can be represented by an *and-tree*, and this is defined implicitly in our next definition.

Definition 2.5 (Defeasible derivation tree) *Let a* DLP $\mathcal{P}$ *and a literal* h *be given. A* defeasible derivation tree T *for* h *from* $\mathcal{P}$*, is a finite, rooted tree (strictly speaking, an* and-tree*), where all nodes are labeled with literals, satisfying the following conditions:*

1) The root node of T *is labeled with (a ground instance of)* h*.*

2) For each node N *in* T *that is labeled with the literal* L*, there is a strict or defeasible rule with head* L_0 *and body* $\{L_1, \ldots, L_k\}$ *in* $\mathcal{P}$*, such that* $L = L_0\sigma$ *for some ground variable substitution* σ*, and the node* N *has exactly* k *children nodes, which are labeled with* $L_1\sigma, \ldots, L_k\sigma$*, respectively.*

If a defeasible derivation for h from $\mathcal{P}$ exists via some derivation tree T (as defined in Definition 2.5), then we will denote this by $\mathcal{P} \vdash_T h$ or simply $\mathcal{P} \vdash h$. Our notion of defeasible derivation tree will be used for the reformulation of specificity in Section 3.

Example 2.6 *Consider the* DLP *of Example 2.4. The literal* $\sim kick(oli)$ *has a defeasible derivation with the instantiated strict rule and the fact in*

$$\left\{ \begin{array}{l} \sim kick(oli) \leftarrow goalie(oli) \\ goalie(oli) \end{array} \right\},$$

and the literal $kick(oli)$ *has a defeasible derivation with*

$$\left\{ \begin{array}{l} kick(oli) \prec player(oli) \\ player(oli) \leftarrow goalie(oli) \\ goalie(oli) \end{array} \right\}.$$

Observe that the literal $\sim kick(diego)$ *has the defeasible derivation:*

$$\left\{ \begin{array}{c} \sim kick(diego) \prec libero(diego) \\ libero(diego) \end{array} \right\}, \qquad [1]$$

whereas $kick(diego)$ *has two defeasible derivations:*

$$\left\{ \begin{array}{c} kick(diego) \prec player(diego) \\ player(diego) \leftarrow libero(diego) \\ libero(diego) \end{array} \right\} \qquad [2]$$

and

$$\left\{ \begin{array}{c} kick(diego) -\!\prec libero(diego), eager(diego) \\ libero(diego) \\ eager(diego) \end{array} \right\}. \qquad [3]$$

Figure 1 shows the corresponding derivation trees of the last three defeasible derivations. In the derivation trees, simple lines denote applications of defeasible rules, while arrows will denote applications of strict rules.

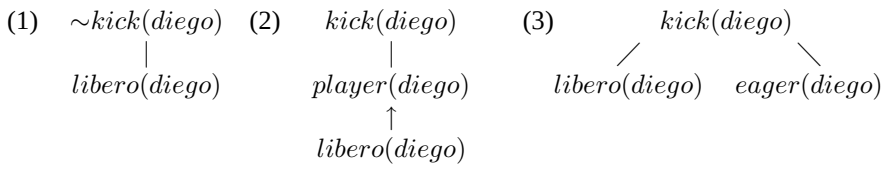

Figure 1. *Derivation trees for Example 2.6.*

Given the DLP of Example 2.4, in Example 2.6 we have just shown that it is possible to have defeasible derivations for two contradictory literals. Thus, a DLP may represent contradictory information. A defeasible logic program $\mathcal{P}$ is *contradictory* (written $\mathcal{P} \vdash \bot$) iff it is possible to defeasibly derive from $\mathcal{P}$ a pair of complementary literals wrt. strong negation, i. e. $\mathcal{P} \vdash p$ and $\mathcal{P} \vdash \sim p$ for some atom p. We will assume that in every DLP $\mathcal{P}$ the set Π is non-contradictory. Otherwise problems as in extended logic programming will happen, and the corresponding analysis of the consequences has been done elsewhere [ALF 96, GEL 90a].

2.3. *Arguments*

The central notion of the DeLP formalism is the notion of an *argument* [SIM 92]. Informally, an argument is a minimal and non-contradictory set of rules used to derive a conclusion. In DeLP, answers to queries will be supported by an argument. The formal definition follows.

Definition 2.7 (Argument) *Let h be a literal and $\mathcal{P} = (\Pi, \Delta)$ be a* DLP. *An* argument *$\mathcal{A}$ for a literal h, also denoted $\langle \mathcal{A}, h \rangle$, is a subset of (ground) instances of defeasible rules of Δ, such that:*

1) there exists a defeasible derivation for h from $\Pi \cup \mathcal{A}$,

2) $\Pi \cup \mathcal{A}$ is non-contradictory, and

3) $\mathcal{A}$ is minimal wrt. set inclusion (i. e., there is no $\mathcal{A}' \subset \mathcal{A}$ such that $\mathcal{A}'$ satisfi es condition 1).

The literal h will also be called the *conclusion* supported by $\mathcal{A}$. An argument $\langle \mathcal{B}, q \rangle$ is a *subargument* of $\langle \mathcal{A}, h \rangle$ iff $\mathcal{B} \subseteq \mathcal{A}$. Note that strict rules are not part of an argument. Observe also that condition 2 of the previous definition prevents the occurrence of "self-defeating" arguments [POL 91].

Example 2.8 *Using the* DLP *of Example 2.4, the literal* $\sim kick(diego)$ *has the argument*

$$\mathcal{A}_1 = \{\ \sim kick(diego) \prec libero(diego)\ \}$$

and the literal $kick(diego)$ *has two arguments:*

$$\mathcal{A}_2 = \{\ kick(diego) \prec player(diego)\ \}$$

and

$$\mathcal{A}_3 = \{\ kick(diego) \prec libero(diego), eager(diego)\ \}$$

The literal $\sim kick(oli)$ *has a derivation formed only by strict rules, so* $\mathcal{A} = \emptyset$ *is an argument for* $\sim kick(oli)$*. Observe that although the set* $\mathcal{B} = \{kick(oli) \prec player(oli), goalie(oli)\}$ *is a defeasible derivation for* $kick(oli)$ *(Example 2.6), there is no argument for this literal because* $\Pi \cup \mathcal{B}$ *is contradictory.*

Given an argument $\mathcal{A}$ for a literal q, other arguments that contradict $\mathcal{A}$ (called *rebuttals* or *counterarguments*) could exist. We say that $\langle \mathcal{A}_1, h_1 \rangle$ *counterargues* $\langle \mathcal{A}_2, h_2 \rangle$ *at a literal* h iff there exists a subargument $\langle \mathcal{A}, h \rangle$ of $\langle \mathcal{A}_2, h_2 \rangle$ such that the set $\Pi \cup \{h_1, h\}$ is contradictory. Therefore, a comparison criterion among arguments is needed. One will be introduced in the next section: generalized specificity. Based on such criterion, the following notion can be introduced.

Definition 2.9 (Defeater [SIM 94]**)** *An argument* $\langle \mathcal{A}_1, h_1 \rangle$ defeats $\langle \mathcal{A}_2, h_2 \rangle$ at literal h *iff there exists a subargument* $\langle \mathcal{A}, h \rangle$ *of* $\langle \mathcal{A}_2, h_2 \rangle$ *such that* $\langle \mathcal{A}_1, h_1 \rangle$ *counterargues* $\langle \mathcal{A}_2, h_2 \rangle$ *at* h*, and one of the following conditions hold:*

1) $\langle \mathcal{A}_1, h_1 \rangle$ *is "better" (wrt. given preference criterion) than* $\langle \mathcal{A}, h \rangle$*; then* $\langle \mathcal{A}_1, h_1 \rangle$ *is a* proper *defeater of* $\langle \mathcal{A}_2, h_2 \rangle$*; or*

2) $\langle \mathcal{A}_1, h_1 \rangle$ *is unrelated by the given preference order to* $\langle \mathcal{A}, h \rangle$*; then* $\langle \mathcal{A}_1, h_1 \rangle$ *is a* blocking *defeater of* $\langle \mathcal{A}_2, h_2 \rangle$.

It is interesting to note that an argument that does not involve defeasible rules *cannot* be defeated. To understand why, assume that $\mathcal{A}_2 = \emptyset$ is an argument for h_2 defeated by $\mathcal{A}_1$ for h_1. Since $\mathcal{A}_2 = \emptyset$, the only existing subarguments of $\mathcal{A}_2$ are also empty arguments $\langle \emptyset, q \rangle$ (i. e. $\Pi \vdash q$). Therefore $\mathcal{A}_1$ counterargues $\mathcal{A}_2$ at some q, i. e. $\Pi \cup \{h_1, q\} \vdash \bot$. Since $\Pi \vdash q$, it follows that $\Pi \cup \{h_1\} \vdash \bot$. But this implies that $\mathcal{A}_1$ does not satisfy condition 2 of Definition 2.7 (contradiction). Note that $\mathcal{A}_1$ must be non-empty, because otherwise Π itself would be contradictory.

In DeLP, a literal q will be considered as *warranted* (or ultimately accepted), if the supporting argument for it is ultimately not defeated. In order to answer the question whether $\mathcal{A}$ is a non-defeated argument, counterarguments that could be *defeaters* for $\mathcal{A}$ are considered. Since defeaters are arguments, there may exist defeaters for the defeaters, and so on. In DeLP a complete dialectical analysis is performed constructing a tree of arguments, called *dialectical tree*, where every node (except the root) is a defeater for its father. In the rest of the paper, we will focus on how conflicting arguments are to be compared, i. e. the relationship between an argument and a defeater. The reader interested in details on the whole dialectical process is referred to [GAR 97, GAR 98, SIM 94].

3. An Inherent Criterion for Comparing Arguments

3.1. *Specificity*

We will formally define a particular criterion called *generalized specifi city* which allows to discriminate between two conflicting arguments. The next definition characterizes the specificity criterion, defined first in [LOU 87, POO 85] and extended later to be used in the defeasible argumentation formalism of [SIM 94, SIM 92]. Here, it is adapted to fit in the DeLP framework. Intuitively, this notion of specificity favors two aspects in an argument: it prefers an argument with greater information content or with less use of defeasible information. In other words, an argument will be deemed better than another if it is *more precise* or *more concise*.

Definition 3.1 (Generalized specificity) *Let $\mathcal{P} = (\Pi, \Delta)$ be a* DLP*, Π_G be the set of all strict rules in Π which are not facts, and $\mathcal{F}$ be the set of all literals that have a defeasible derivation from $\mathcal{P}$. Then, $\langle \mathcal{A}_1, h_1 \rangle$ is* more specific than $\langle \mathcal{A}_2, h_2 \rangle$ *(written $\langle \mathcal{A}_1, h_1 \rangle \succeq \langle \mathcal{A}_2, h_2 \rangle$) iff for all $H \subseteq \mathcal{F}$ it holds:*

$$\Pi_G \cup H \cup \mathcal{A}_1 \vdash h_1 \text{ and } \Pi_G \cup H \nvdash h_1 \text{ imply } \Pi_G \cup H \cup \mathcal{A}_2 \vdash h_2.$$

According to [POO 85], we defi ne: $\langle A_1, h_1 \rangle$ is strictly more specific *than $\langle A_2, h_2 \rangle$ (written $\langle A_1, h_1 \rangle \succ \langle A_2, h_2 \rangle$) iff $\langle A_1, h_1 \rangle \succeq \langle A_2, h_2 \rangle$ and $\langle A_2, h_2 \rangle \not\succeq \langle A_1, h_1 \rangle$. This means,*

1) for all sets $H \subseteq \mathcal{F}$ it holds that, if $\Pi_G \cup H \cup \mathcal{A}_1 \vdash h_1$ and $\Pi_G \cup H \nvdash h_1$, then $\Pi_G \cup H \cup \mathcal{A}_2 \vdash h_2$, and

2) there exists a set $H' \subseteq \mathcal{F}$ such that $\Pi_G \cup H' \cup \mathcal{A}_2 \vdash h_2$ and $\Pi_G \cup H' \nvdash h_2$, and $\Pi_G \cup H \cup \mathcal{A}_1 \nvdash h_1$.

In the definition above the set Π_G does not contain facts, so the condition $\Pi_G \cup H \cup \mathcal{A}_1 \vdash h_1$ will hold only with some particular non-empty set H. Remember that we do not consider presumptions in this context. We say that H *activates* $\mathcal{A}_1$. The expression $\Pi_G \cup H \nvdash h_1$ is called the *non-triviality condition*, because it stresses

the need for use of the set $\mathcal{A}_1$ for deriving h_1. Hence, Definition 3.1 may be read as: $\langle \mathcal{A}_1, h_1 \rangle$ is more specific than $\langle \mathcal{A}_2, h_2 \rangle$ iff for each set H that non-trivially activates $\mathcal{A}_1$, the same set H activates $\mathcal{A}_2$.

Continuing with Example 2.8, argument $\langle \mathcal{A}_1, \sim kick(diego) \rangle$ is *strictly more specific* than $\langle \mathcal{A}_2, kick(diego) \rangle$ (see below), because $\langle \mathcal{A}_1, \sim kick(diego) \rangle$ does not use the strict rule $player(X) \leftarrow libero(X)$ and hence is more direct. Observe that condition 1 in Definition 3.1 holds: every set H that activates $\langle \mathcal{A}_1, \sim kick(diego) \rangle$ also activates $\langle \mathcal{A}_2, kick(diego) \rangle$. However, the set $H' = \{player(diego)\}$ activates $\langle \mathcal{A}_2, kick(diego) \rangle$, but does not activate $\langle \mathcal{A}_1, \sim kick(diego) \rangle$.

$$\mathcal{A}_1 = \left\{ \ \sim kick(diego) \prec libero(diego) \ \right\}$$

$$\mathcal{A}_2 = \left\{ \ kick(diego) \prec player(diego) \ \right\}$$

On the other hand, the argument $\langle \mathcal{A}_3, kick(diego) \rangle$ (see below) will be regarded as strictly more specific than $\langle \mathcal{A}_1, \sim kick(diego) \rangle$, because $\langle \mathcal{A}_3, kick(diego) \rangle$ is based on more information ($libero$ and $eager$). Again, the condition holds and the set $H'' = \{libero(diego)\}$ is enough to activate $\langle \mathcal{A}_1, \sim kick(diego) \rangle$ but does not activate $\langle \mathcal{A}_3, kick(diego) \rangle$.

$$\mathcal{A}_3 = \left\{ \ kick(diego) \prec libero(diego), eager(diego) \ \right\}$$

Thus, in summary we have $\langle \mathcal{A}_3, kick(diego) \rangle \succ \langle \mathcal{A}_1, \sim kick(diego) \rangle \succ \langle \mathcal{A}_2, kick(diego) \rangle$. We also have $\langle \mathcal{A}_3, kick(diego) \rangle \succ \langle \mathcal{A}_2, kick(diego) \rangle$. This means, in principle, we can compare also arguments with non-contradictory (even identical) conclusions wrt. (generalized) specificity. Specificity is a comparison criterion that is independent of the notion of counterargument or defeat. Nevertheless, in the DeLP framework comparing arguments is triggered only when contradictory conclusions are arrived (see Definition 2.9).

The following example shows why comparing only a pair of rules instead of complete arguments may sometimes be unsatisfactory. Consider the following program:

$$\left\{ \begin{array}{ll} s(X) \leftarrow q(X) & \\ q(a) & \\ p(X) \prec q(X) & (r_1) \\ \sim p(X) \prec q(X), s(X) & (r_2) \end{array} \right\}$$

The rule $r_2 = \sim p(X) \prec q(X), s(X)$ is using more information than $r_1 = p(X) \prec q(X)$. Thus, in a system with priorities over rules such as [COV 88, NUT 94], it is expected to have that r_2 is preferred to r_1 (written $r_2 > r_1$). Therefore, in such a system the conclusion $\sim p(a)$ will be preferred over $p(a)$. But in this case, it is not true that $\sim p(a)$ is using more information, because the rule $s(X) \leftarrow q(X)$ establishes

a strict connection between $s(X)$ and $q(X)$ (every $q(X)$ is an $s(X)$). In DeLP, the argument $\mathcal{A} = \{(\sim p(a) \prec q(a), s(a))\}$[1] for $\sim p(a)$ is not *strictly* more specific than the argument $\mathcal{B} = \{p(a) \prec q(a)\}$ for $p(a)$, and vice versa. However, it holds $\langle \mathcal{A}, \sim p(a) \rangle$ is more specific than $\langle \mathcal{B}, p(a) \rangle$, and vice versa.

In some systems [ANT 00, MAH 98] the rules r_1 and r_2 can be left unrelated wrt. superiority, and then achieve the desired result. But note that, if the rule $s(X) \leftarrow q(X)$ is replaced with the fact $s(a)$ (i. e., there is no longer a connection between $s(X)$ and $q(X)$), then using the generalized specificity notion (Definition 3.1), the argument $\mathcal{A}$ will be strictly more specific than $\mathcal{B}$. However, in a system with fixed priorities over rules this automatic change in the behavior of the system is not possible. The priority (or superiority) relation has to be changed to produce the expected result.

3.2. *Arguments and Pruning*

The following example reveals that in Definition 3.1—and hence also in [POO 85, SIM 92]—we cannot restrict our attention to derivations which only make use of the defeasible rules in the given arguments. Therefore, we will introduce the concept of *pruning* in Definition 3.3, in order to establish equivalence between the original definition of specificity (Definition 3.1) and the new characterizations of specificity by activation sets as in Section 3.3 or by path sets as in Section 3.4.

From a computational point of view, pruning complicates the procedure for computing specificity. But the reason for this is that we consider a very general setting in this paper, because we admit

1) more than one antecedent in rules, i. e. bodies containing more than one (possibly negative) literal, and

2) (posssibly) non-empty sets of background knowledge, i. e. strict rules, not only facts.

In the literature, often restricted cases are considered only: antecedents are always singletons in [GEL 90b], no background knowledge is allowed in [DUN 96], and both restrictions are present in [BEN 97]. Since we consider the general case here, Example 3.2 can be formulated. It shows that for computing specificity we cannot concentrate on derivations using the rules of the given argument only. If we do this, then things will become computationally simpler. Therefore, in the following Section 3.3 we will also introduce an alternative definition of specificity that will fix the problem shown in the Example 3.2.

1. We use parentheses just for improving the readability of the set of rules.

Example 3.2 *Let us consider the following program:*

$$\left\{\begin{array}{l} x \prec a, b, c \\ a \prec d \\ b \prec e \\ \sim x \prec a, b \\ c \\ d \\ e \\ x \leftarrow a, f \\ f \prec e \end{array}\right\}$$

From this program, the arguments $\langle \mathcal{A}, x\rangle$ *and* $\langle \mathcal{B}, \sim x\rangle$ *with*

$$\mathcal{A} = \{(x \prec a, b, c), (a \prec d), (b \prec e)\}$$

and

$$\mathcal{B} = \{(\sim x \prec a, b), (a \prec d), (b \prec e)\}$$

can be obtained.

By Definition 3.1, it holds that $\langle \mathcal{A}, x\rangle$ is not more specific than $\langle \mathcal{B}, \sim x\rangle$, since the strict rule $x \leftarrow a, f$ together with $\mathcal{A}$ and $H = \{d, f\}$ non-trivially activate x (because the defeasible rule $(a \prec d) \in \mathcal{A}$ can be used), but Π_G together with $\mathcal{B}$ and $H = \{d, f\}$ do not activate $\sim x$. Figure 2 shows derivation trees for the arguments $\langle \mathcal{A}, x\rangle$, $\langle \mathcal{B}, \sim x\rangle$ and in addition $\langle \mathcal{C}, x\rangle$ (in this order) where:

$$\mathcal{C} = \{(a \prec d), (f \prec e)\}$$

Observe that for the comparison of the arguments $\langle \mathcal{A}, x\rangle$ and $\langle \mathcal{B}, \sim x\rangle$ according to Definition 3.1, a derivation which uses another argument set $\mathcal{C}$ has to be taken into consideration, making the activation set $\{d, f\}$ possible. This derivation is shown in Figure 2 (c). Here (and also in the sequel), dotted lines lead to parts of the derivation tree which must not be considered when comparing the arguments $\mathcal{A}$ and $\mathcal{B}$. We obtain these parts of derivation trees by *pruning*, which we define formally in our next definition.

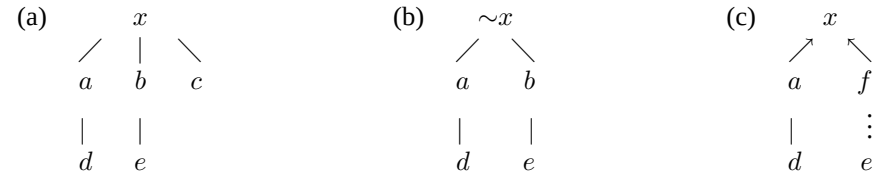

Figure 2. *Derivation trees for Example 3.2.*

Definition 3.3 (Pruned derivation trees) *Let $\langle \mathcal{A}, h\rangle$ be an argument in a program $\mathcal{P} = \Pi \cup \Delta$. Let T be a defeasible derivation tree for h in $\mathcal{P}$, i. e. $\Pi \cup \Delta \vdash_T h$. We defi ne a* derivation tree pruned wrt. the argument $\langle \mathcal{A}, h\rangle$*, denoted $T^{\langle \mathcal{A}, h\rangle}$, as the tree obtained from T by deleting all nodes in T which occur below nodes labeled with head literals of defeasible rules $r \notin \mathcal{A}$ (or instances thereof).*

In order to illustrate Definition 3.3, let us revisit Example 2.4 and Figure 1 (3) again. The derivation tree T in Figure 1 (3) makes only use of the rule $r = (kick(diego) \prec libero(diego), eager(diego))$. Thus, T is already pruned wrt. the argument $\mathcal{A}_3 = \{r\}$. If we prune T wrt. $\mathcal{A}_2 = \{kick(diego) \prec player(diego)\}$, which is also an argument for $h = kick(diego)$, then all nodes below the root have to be deleted, because $r \notin \mathcal{A}_2$. Hence, $T^{\langle \mathcal{A}_2, h\rangle}$ simply consists of one node which is labeled with h. This pruned tree will not be considered when comparing arguments, because it does not involve any defeasible rule. But things are not always that simple (see Example 3.2).

3.3. *Characterization by Activation Sets*

Definition 3.1 suggests to test all subsets $H \subseteq \mathcal{F}$. Hence, if $\mathcal{F}$ contains n elements, there are 2^n sets to be considered. Besides the exponential explosion problem, this definition might be considering sets of literals that are unrelated to the arguments being compared. In addition, computing the set $\mathcal{F}$ is a problem on its own. In this section, we introduce a way of avoiding these problems, which will be continued in the next section.

Definition 3.4 (Argument completion: first version) *Let $\mathcal{P} = (\Pi, \Delta)$ be a* DLP*. A completion of an argument $\mathcal{A}$ for h, denoted $\underline{\mathcal{A}}$, is the set of defeasible and strict rules without facts in $\Pi_G \cup \mathcal{A}$, which are used in a derivation tree T for h that is pruned wrt. $\langle \mathcal{A}, h\rangle$.*

Note that for a particular argument $\langle \mathcal{A}, h\rangle$, there are (possibly) many alternative completions (see Definitions 3.9 and 3.10). In Definition 3.4, the notion of argument completion depends on the notion of pruning (Definition 3.3). But pruning is certainly an expensive operation, because we cannot restrict our attention to derivations which only make use of the defeasible rules in the given arguments only. We must take into consideration also related (pruned) arguments. Therefore, we propose the following alternative definition of argument completion that, from a computational point of view, simplifies and hence improves the first one. This definition is also used in the implementation of the DLP system (described in [GAR 97]).

Definition 3.5 (Argument completion: alternative version) *Let $\mathcal{P}$ be a* DLP*, and let $\langle \mathcal{A}, h\rangle$ be an argument in $\mathcal{P}$. Let T be a derivation tree for h that does not make use of any defeasible rule $r \notin \mathcal{A}$. Then a completion of $\langle \mathcal{A}, h\rangle$ is the set of defeasible and strict rules (without facts) in $\Pi_G \cup \mathcal{A}$ which are used in T.*

Example 3.6 *Let us consider the following program:*

$$\left\{\begin{array}{l} h \leftarrow a \\ a \prec b, c \\ b \leftarrow d \\ b \prec e \\ d \\ e \\ c \end{array}\right\}$$

For this program, we have the argument $\langle \mathcal{A}, h\rangle$ with $\mathcal{A} = \{a \prec b, c\}$. One possible argument completion for $\langle \mathcal{A}, h\rangle$ is $\{(h \leftarrow a), (a \prec b, c), (b \leftarrow d)\}$ according to both definitions of completion. But there is also another derivation for h making use of the rules in $\{h \leftarrow a, a \prec b, c, b \prec e\}$. Since the last defeasible rule in this set does not belong to the original argument $\mathcal{A}$, another argument completion (according to Definition 3.4) after pruning, i. e. deleting this rule, is $\underline{\mathcal{A}} = \{h \leftarrow a, a \prec b, c\}$.

Note that an argument completion $\underline{\mathcal{A}}$ does not contain the facts used in the construction of the defeasible tree. The reason for that will become clear later. The set of ground literals of $\underline{\mathcal{A}}$, denoted $Lit(\underline{\mathcal{A}})$, will be the set of all ground literals that occur in the body or head of every rule in $\underline{\mathcal{A}}$. Considering Example 3.6, it follows that $Lit(\underline{\mathcal{A}}) = \{a, b, c, d, h\}$.

Definition 3.7 (Activation set) *Let $\underline{\mathcal{A}}$ be a completed argument, and $Lit(\underline{\mathcal{A}})$ the corresponding set of literals. A set $U \subseteq Lit(\underline{\mathcal{A}})$ is an* activation set *for $\underline{\mathcal{A}}$, if $U \cup \underline{\mathcal{A}} \vdash h$, and U is minimal with respect to set inclusion (i. e., $\nexists U' \subset U$ such that $U' \cup \underline{\mathcal{A}} \vdash h$). We will call Act-sets($\underline{\mathcal{A}}$) the set of all activation sets for $\underline{\mathcal{A}}$.*

Definition 3.8 (Non-trivial activation set) *Let $\underline{\mathcal{A}}$ be a completed argument, and $Lit(\underline{\mathcal{A}})$ the corresponding set of literals. A set $U \subseteq Lit(\underline{\mathcal{A}})$ is called a* non-trivial activation set *for $\underline{\mathcal{A}}$, if U is an activation set for $\underline{\mathcal{A}}$ and $U \cup \Pi_G \nvdash h$. We will call NTAct-sets($\underline{\mathcal{A}}$) the set of all the non-trivial activation sets for $\underline{\mathcal{A}}$.*

Figure 3 shows an algorithm to compute all non-trivial activation sets for a completed argument $\underline{\mathcal{A}}$ for h. In order to avoid trivial activation sets we only check whether a defeasible rule has been used. Note that the first activation set for $\mathcal{A}$ is h itself, and it is also a trivial one. As can be seen from the algorithm, the set of activation sets of an argument is easy to compute, just parsing the completed argument once. Specificity can thus be defined in a form such that we only need to consider the non-trivial activation sets.

Definition 3.9 (Specificity revisited: preliminary version) *Let $\langle \mathcal{A}_1, h_1\rangle$ and $\langle \mathcal{A}_2, h_2\rangle$ be two arguments, and $\underline{\mathcal{A}_1}$ and $\underline{\mathcal{A}_2}$ be completed arguments for $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively. We say that $\langle \mathcal{A}_1, h_1\rangle$ is strictly more specific than $\langle \mathcal{A}_2, h_2\rangle$ iff*

Input: a completed argument $\underline{\mathcal{A}}$ for h.
Output: NTAct-sets($\underline{\mathcal{A}}$)

1) A stack S is initialized with the pair $(\{h\}, trivial)$.

2) NTAct-sets($\underline{\mathcal{A}}$) is initialized empty.

3) Repeat until S is empty:

a) Select the first pair (N,$type$) in S and remove it from the stack.

b) If $type$ is *non-trivial* then add N to NTAct-sets($\underline{\mathcal{A}}$).

c) The element N will be formed by a set of literals $l_1, \ldots, l_k$. For each literal $l_i \in N$ that is a head of a rule r in $\underline{\mathcal{A}}$, with no empty body, create a new activation set N_i replacing l_i with the literals in the body of r. The type of N_i is *trivial* only if the type of N is *trivial* and r is a strict rule. Otherwise the type of N_i is *non-trivial*. Thus, for every literal l_i in N a new activation set can be created.

d) The new activation sets N_i, that were not previously expanded, are added to the top of S.

4) Return NTAct-sets($\underline{\mathcal{A}}$)

Figure 3. *Computing non-trivial activation sets.*

1) for all sets $U \in$ NTAct-sets($\underline{\mathcal{A}}$) it holds $\Pi_G \cup U \cup \mathcal{A}_2 \vdash h_2$, and

2) there exists a set $U' \in$ NTAct-sets($\underline{\mathcal{B}}$) such that $\Pi_G \cup U' \cup \mathcal{A}_1 \nvdash h_1$.

In general, there is no unique $\underline{\mathcal{A}}$ for a given argument $\mathcal{A}$ for a literal h, since different rules in Π_G can be used to prove the body literals of defeasible rules in $\mathcal{A}$. Nevertheless, the difference between two argument completions $\underline{\mathcal{A}}_1$ and $\underline{\mathcal{A}}_2$ for an argument $\mathcal{A}$ lies only in the used strict rules for the defeasible derivation. Definition 3.9 is equivalent to Definition 3.1 only if there is a unique completion for each argument. However, it can be improved in terms of the sets defined below which consider every possible completion for an argument. In the following, $ArgComp(\underline{\mathcal{A}})$ denotes the set of all argument completions of $\mathcal{A}$. We define:

$$\text{Act-sets}(\mathcal{A}) = \bigcup_{\underline{\mathcal{A}} \in ArgComp(\underline{\mathcal{A}})} \text{Act-sets}(\underline{\mathcal{A}})$$

$$\text{NTAct-sets}(\mathcal{A}) = \bigcup_{\underline{\mathcal{A}} \in ArgComp(\underline{\mathcal{A}})} \text{NTAct-sets}(\underline{\mathcal{A}})$$

Definition 3.10 (Specificity revisited: final version) *Let $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_2, h_2 \rangle$ be two arguments. We say that $\langle \mathcal{A}_1, h_1 \rangle$ is* strictly more specific than *$\langle \mathcal{A}_2, h_2 \rangle$ (written $\langle \mathcal{A}_1, h_1 \rangle \sqsupset \langle \mathcal{A}_2, h_2 \rangle$ in this case) iff*

1) for all sets $U \in$ NTAct-sets($\mathcal{A}_1$) it holds $\Pi_G \cup U \cup \mathcal{A}_2 \vdash h_2$, and

2) there exists a set $U' \in$ NTAct-sets($\mathcal{A}_2$) such that $\Pi_G \cup U' \cup \mathcal{A}_1 \nvdash h_1$.

Let us come back to Example 3.2. According to Definition 3.10 with the alternative definition of argument completion (Definition 3.5), i. e. without taking derivations into account which use defeasible rules $r \notin \mathcal{A}$, we have that $\langle \mathcal{A}, x\rangle$ is strictly more specific than $\langle \mathcal{B}, \sim x\rangle$, since $\{d, f\}$ is not considered as an activation set for $\langle \mathcal{A}, x\rangle$ (because f does not occur in the derivation which uses only the defeasible rules from $\mathcal{A}$). This means that $\langle \mathcal{A}, x\rangle$ is more specific than $\langle \mathcal{B}, \sim x\rangle$. But $\langle \mathcal{B}, \sim x\rangle$ is not more specific than $\langle \mathcal{A}, x\rangle$, because $\{a, b\}$ non-trivially activates $\sim x$, but not x.

However, according to Definition 3.10 with the original version of argument completion (Definition 3.4), it turns out that $\langle \mathcal{A}, x\rangle$ is *not* strictly more specific than $\langle \mathcal{B}, \sim x\rangle$. The problem is that there are two arguments for x, namely $\mathcal{A}$ and $\mathcal{C} = \{(a \prec d), (f \prec e)\}$, which makes the activation set $\{d, f\}$ possible. We come to the same conclusion when applying Definition 3.1. In fact, if we adopt Definition 3.4 for argument completion, then we have the following equivalence:

Theorem 3.11 *Let $\langle \mathcal{A}_1, h_1\rangle$ and $\langle \mathcal{A}_2, h_2\rangle$ be two arguments in a program $\mathcal{P}$. Then, $\langle \mathcal{A}_1, h_1\rangle \sqsupset \langle \mathcal{A}_2, h_2\rangle$ iff $\langle \mathcal{A}_1, h_1\rangle \succ \langle \mathcal{A}_2, h_2\rangle$.*

Proof: In the following, we write $\langle \mathcal{A}_1, h_1\rangle \sqsupseteq \langle \mathcal{A}_2, h_2\rangle$ iff (at least) condition 1 of Definition 3.10 holds. Obviously, in such a case condition 2 of Definition 3.10 is equivalent to $\langle \mathcal{A}_2, h_2\rangle \not\sqsupseteq \langle \mathcal{A}_1, h_1\rangle$. Therefore, we have $\langle \mathcal{A}_1, h_1\rangle \sqsupset \langle \mathcal{A}_2, h_2\rangle$ iff $\langle \mathcal{A}_1, h_1\rangle \sqsupseteq \langle \mathcal{A}_2, h_2\rangle$ and $\langle \mathcal{A}_2, h_2\rangle \not\sqsupseteq \langle \mathcal{A}_1, h_1\rangle$. Since the relationship between $\succ$ and $\succeq$ is defined analogously in Definition 3.1, we only have to show that $\langle \mathcal{A}_1, h_1\rangle \sqsupseteq \langle \mathcal{A}_2, h_2\rangle$ iff $\langle \mathcal{A}_1, h_1\rangle \succeq \langle \mathcal{A}_2, h_2\rangle$, because this implies the conjecture.

Let us first prove the direction from left to right of this statement. Thus, by hypothesis it holds that $\Pi_G \cup H \cup \mathcal{A}_1 \vdash h_1$ for some set of possible facts H with $\Pi_G \cup H \nvdash h_1$ (non-triviality condition). This means there is a derivation T with $\Pi_G \cup H \cup \mathcal{A}_1 \vdash_T h_1$. Since Π_G does not contain any facts, the leaves in T must be labeled with literals from H. Because of $H \subseteq \mathcal{F}$ (the set of possible facts), there must exist a derivation T' for h_1 from $\mathcal{P}$, that is identical with T, but completed with additional subderivations below those leaves in T which are not facts in $\mathcal{P}$ and afterwards pruned wrt. $\mathcal{A}_1$.

Let now $\underline{\mathcal{A}_1}$ be the completed argument wrt. the derivation tree T' (according to Definition 3.4). Clearly, $H \subseteq Lit(\underline{\mathcal{A}_1})$ and $H \cup \underline{\mathcal{A}_1} \vdash h_1$. Let now U be a minimal subset of H such that $U \cup \underline{\mathcal{A}_1} \vdash h_1$. By Definition 3.8, U is a non-trivial activation set wrt. $\underline{\mathcal{A}_1}$. Therefore, it holds $U \in$ NTAct-sets($\mathcal{A}_1$). But then, the precondition $\langle \mathcal{A}_1, h_1\rangle \sqsupseteq \langle \mathcal{A}_2, h_2\rangle$ implies $\Pi_G \cup U \cup \mathcal{A}_2 \vdash h_2$ by Definition 3.10. Because of $H \supseteq U$, it follows $\Pi_G \cup H \cup \mathcal{A}_2 \vdash h_2$ and thus $\langle \mathcal{A}_1, h_1\rangle \succeq \langle \mathcal{A}_2, h_2\rangle$. This completes the first part of this proof.

The direction from right to left ($\langle \mathcal{A}_1, h_1\rangle \succeq \langle \mathcal{A}_2, h_2\rangle$ implies $\langle \mathcal{A}_1, h_1\rangle \sqsupseteq \langle \mathcal{A}_2, h_2\rangle$) is more or less straightforward. Let $U \in$ NTAct-sets($\mathcal{A}_1$) be a non-trivial activation set. Then by Definition 3.8, there is a completed argument $\underline{\mathcal{A}_1}$, such that $U \subseteq Lit(\underline{\mathcal{A}_1})$ and $U \cup \underline{\mathcal{A}_1} \vdash h_1$. According to Definition 3.4, $\underline{\mathcal{A}_1}$ stems from a derivation tree T^* that is pruned wrt. $\mathcal{A}_1$. This implies $\underline{\mathcal{A}_1} \subseteq \Pi_G \cup \mathcal{A}_1$. Hence, it holds $\Pi_G \cup U \cup \mathcal{A}_1 \vdash h_1$. Since U is a non-trivial activation set by hypothesis, it follows

$\Pi_G \cup U \cup \mathcal{A}_2 \vdash h_2$ because of the precondition $\langle \mathcal{A}_1, h_1 \rangle \succeq \langle \mathcal{A}_2, h_2 \rangle$ (see Definition 3.1). Thus, finally we have $\langle \mathcal{A}_1, h_1 \rangle \sqsupseteq \langle \mathcal{A}_2, h_2 \rangle$. This completes the second part of the proof. ■

3.4. *Characterization by Path Sets*

In the previous section, we expressed specificity by means of activation sets. In this section, we will go one step further by defining specificity via the comparison of (sets of) derivations. For this, we will identify each defeasible derivation tree with its sets of paths in the tree.

Let N be a leaf node in a (possibly pruned) derivation tree T. We define the *path* in T through N as the set consisting of the literal labeling N, together with all literals labeling its ancestors (except the root node). Let $Paths(T)$ be the set of all paths in T through all leaf nodes N.

Example 3.12 *The path sets for the derivation trees in Figure 1, which are already pruned wrt. the corresponding arguments, are (1)* $\{\{libero(diego)\}\}$, *(2)* $\{\{player(diego), libero(diego)\}\}$, *and (3)* $\{\{libero(diego)\}, \{eager(diego)\}\}$, *respectively. Consider example 2.4 and the tree* T *in Figure 1 (3) once again. Since* $T^{\langle \mathcal{A}_2, h \rangle}$ *simply consists of one node—the root node—which is labeled with* h, *it holds* $Paths(T^{\langle \mathcal{A}_2, h \rangle}) = \{\emptyset\}$.

With this notion of paths, we are able to give a (preliminary) syntactic definition of specificity as follows, by introducing the relation $\trianglerighteq$. We will see later (in Theorem 3.15) that $\trianglerighteq$ and $\succeq$ are equivalent if the arguments involved in the comparison correspond to exactly one derivation tree.

Definition 3.13 *Let* T_1 *and* T_2 *be two derivation trees. We define* $T_1 \trianglerighteq T_2$ *iff for all* $t_2 \in Paths(T_2)$ *there exists a path* $t_1 \in Paths(T_1)$ *such that* $t_1 \subseteq t_2$.

As already observed in the previous section, an argument cannot always be identified with one unique derivation or completed argument, but with a set of those. Therefore, we will take this into account in our next definition.

Definition 3.14 (Syntactic criterion) *Let* $\langle \mathcal{A}_1, h_1 \rangle$ *and* $\langle \mathcal{A}_2, h_2 \rangle$ *be two arguments in a program* $\mathcal{P}$. *Then* $\langle \mathcal{A}_1, h_1 \rangle \geq \langle \mathcal{A}_2, h_2 \rangle$ *iff for all derivation trees* T_1 *for* h_1 *pruned wrt.* $\mathcal{A}_1$ *there is a tree* T_2 *for* h_2 *pruned wrt.* $\mathcal{A}_2$ *such that* $T_1 \trianglerighteq T_2$.

Now, we are able to state yet another formulation of specificity by means of the relation $\geq$ in the subsequent theorem. It gives us a syntactic characterization of specificity without guessing sets of possible facts $H \subseteq \mathcal{F}$.

Theorem 3.15 *Let $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_2, h_2 \rangle$ be two arguments in a program $\mathcal{P}$. Then, $\langle \mathcal{A}_1, h_1 \rangle \geq \langle \mathcal{A}_2, h_2 \rangle$ implies $\langle \mathcal{A}_1, h_1 \rangle \succeq \langle \mathcal{A}_2, h_2 \rangle$. If Π_G is empty, then also the converse holds: $\langle \mathcal{A}_1, h_1 \rangle \succeq \langle \mathcal{A}_2, h_2 \rangle$ implies $\langle \mathcal{A}_1, h_1 \rangle \geq \langle \mathcal{A}_2, h_2 \rangle$.*

Proof: Let us first prove the first part of the statement. Thus, by hypothesis it holds that $\Pi_G \cup H \cup \mathcal{A}_1 \vdash h_1$ for some set of possible facts H with $\Pi_G \cup H \nvdash h_1$ (non-triviality condition). This means there is a derivation T_1 with $\Pi_G \cup H \cup \mathcal{A}_1 \vdash_{T_1} h_1$. Since Π_G does not contain any facts, the leaves in T_1 must be labeled with literals from H. Because of $H \subseteq \mathcal{F}$ (the set of possible facts), there must exist a derivation T_1' for h_1 from $\mathcal{P}$, that is identical with T_1, but completed with additional subderivations below those leaves in T_1 which are not facts in $\mathcal{P}$ in $\mathcal{A}_1$, and afterwards pruned wrt. $\mathcal{A}_1$.

Now, by precondition, there is a derivation T_2' for h_2 pruned wrt. $\mathcal{A}_2$ such that $T_1' \trianglerighteq T_2'$. Since by hypothesis the activation set H for $\langle \mathcal{A}_1, h_1 \rangle$ is non-trivial, for each path $t \in Paths(T_2')$, there must be a literal $L \in H$, since otherwise there would be a path t^* in $Paths(T_2')$, such that no element of H occurs in t^*, but this contradicts $T_1' \trianglerighteq T_2'$. Now we delete all subderivations below nodes labeled with a literal $L \in H$. Obviously, the obtained tree T_2 is a derivation tree, satisfying $\Pi_G \cup H \cup \mathcal{A}_2 \vdash_{T_2} h_2$. Hence, it holds $\Pi_G \cup H \cup \mathcal{A}_2 \vdash h_2$. This completes the first part of the proof.

In order to show the second part, we first notice that since an argument $\mathcal{A}$ must be minimal (according to condition 3 of Definition 2.7), for each literal L, there can be at most one defeasible rule with head L in $\mathcal{A}$. Furthermore, since Π_G is empty by precondition in this case, it follows, that every argument $\langle \mathcal{A}, h \rangle$ corresponds to exactly one derivation tree T. This observation will be helpful in the analysis that follows.

Let T_1 be the derivation for $\langle \mathcal{A}_1, h_1 \rangle$ (pruned wrt. $\mathcal{A}_1$), which exists by hypothesis. The leafs of this tree clearly are labeled with program facts. Let H denote the set of facts among them. Obviously, $\mathcal{A}_1 \cup H \vdash h_1$. It also holds $\Pi_G \cup H \nvdash h_1$, since Π_G is empty and $h_1 \notin H$ (because otherwise h_1 must be a fact in $\mathcal{P}$, which implies that $\mathcal{A}_1$ is not minimal). Therefore, we conclude $\mathcal{A}_2 \cup H \vdash h_2$ because of $\langle \mathcal{A}_1, h_1 \rangle \succeq \langle \mathcal{A}_2, h_2 \rangle$ (by precondition).

Let T_2 be the corresponding derivation (pruned wrt. $\mathcal{A}_2$), which is unique as stated above. Let us assume $T_1 \not\trianglerighteq T_2$. Then there must be a path $t_2 \in Paths(T_2)$ such that for all paths $t_i \in Paths(T_1)$ it holds $t_i \not\supseteq t_2$. This means that in each path t_i there must be (at least) one literal L which is not in t_2. Let H' be the set of all these literals. Clearly, H' is a non-trivial activation set for $\langle \mathcal{A}_1, h_1 \rangle$, since Π_G is empty and H' cannot contain h_1 (because $h_1 \notin t_i$ for all $t_i \in Paths(T_1)$).

Thus by precondition, there must be a derivation T_2' with $\mathcal{A}_2 \cup H' \vdash_{T_2'} h_2$. Now, T_2' can be completed with additional subderivations and pruned wrt. $\mathcal{A}_2$ such that all leaves are facts. However, the obtained tree is different from T_2, because it cannot contain the path t_2. But this contradicts to the fact that derivation trees are uniquely determined. Hence, it holds $T_1 \trianglerighteq T_2$, and finally $\langle \mathcal{A}_1, h_1 \rangle \geq \langle \mathcal{A}_2, h_2 \rangle$. This completes the second part of the proof. ■

Example 3.16 *In order to see the necessity of the restriction in the second part of Theorem 3.15, let us consider the following program:*

$$\{(x \prec a, b), (b \prec c), (\sim x \prec c, d), (d \prec a), a, c, (\sim x \leftarrow b, d)\}$$

For the arguments $\mathcal{A} = \{(x \prec a, b), (b \prec c)\}$ *and* $\mathcal{B} = \{(\sim x \prec c, d), (d \prec a)\}$*, it holds* $\langle \mathcal{A}, x\rangle \succeq \langle \mathcal{B}, \sim x\rangle$*, because* $\{a, b\}$ *and* $\{a, c\}$ *(and their supersets not containing* x*) are the only non-trivial activation sets for* $\langle \mathcal{A}, x\rangle$*, which also activate* $\langle \mathcal{B}, \sim x\rangle$*. However,* $\langle \mathcal{A}, x\rangle \not\succeq \langle \mathcal{B}, \sim x\rangle$*, because there is only one derivation* T_1 *for* x *(shown in Figure 4 (a)), and there are two derivations* T_2 *(shown in Figures 4 (b) and 4 (c)) for* $\sim x$ *(pruned wrt.* $\mathcal{B}$*), but for both of them it holds* $T_1 \not\trianglerighteq T_2$*. In order to see this, note that the literal* c *actually does not belong to the tree in Figure 4 (c), because it has to be pruned wrt. the argument* $\mathcal{B}$*.*

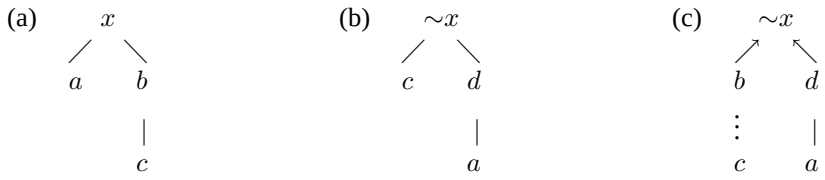

Figure 4. *Derivation trees for Example 3.16.*

3.5. *Summary*

In this section, we have introduced two different characterizations of specificity, namely by activation sets (Section 3.3) and path sets (Section 3.4). In contrast to other approaches, we consider a very general setting here, namely where (i) antecedents of rules may be arbitrary large sets of positive and negative literals, and (ii) mixed rule sets are allowed, i. e. with defeasible and also strict rules (as already mentioned in Section 3.2). Therefore, our proposal can be seen as an extension of several approaches in the literature [AMG 96, BEN 97, DUN 96, GEL 90b, HOR 94].

As we have seen, the original definition of general specificity (Definition 3.1) can be characterized equivalently by activation or path sets, which decreases the computational complexity for specificity. With activation sets, we do not have to try out all of the exponentially many possible activation sets (as already stated in Section 3.3). The same holds for the characterization by path sets, which allows us to restrict our attention to the derivations of the given literal only. Hence, both characterizations yield us criteria that can be implemented in a computationally attractive way.

The complexity for computing specificity in the restricted setting where both restrictions from Section 3.2 are present (antecedents are singletons; no strict background knowledge) can easily be determined for our approach. In this case there is only one derivation for each literal (which can be seen in the proof of Theorem 3.15,

second part), and each derivation is just a linear sequence of literals. Thus, checking our syntactic criterion $\geq$ for the comparison of arguments consists of just one subset test that can be done in polynomial time, i. e. quite efficiently.

Interestingly, the combination of all these linear derivations from above in one graph resembles *defeasible inheritance networks* as defined in [HOR 94]. There are many similarities between reasoning in these networks and our approach. Horty [HOR 94] defines the notion *defeasible inheritability* of paths in a network. According to this definition, paths have to be constructible, non-conflicting, and non-preemptive. These notions loosely correspond to the notions derivable (Definition 2.5) and strictly more general (wrt. specificity, Definition 3.1) in our context. The above-mentioned properties of networks can also be tested in polynomial time.

Besides these obvious relations and similarities, there is one major conceptual difference. Horty's (and others') approaches base their notion of defeasible inheritance on *procedures* that can be applied to defeasible inheritance networks. But this might lead to counterintuitive results. For instance, cyclic networks do not always have an extension, and for cycles with strict rules, special procedures have to be employed, e. g. by computing equivalence classes (see [HOR 94, pp. 125-141]). In contrast to this, our notion of defeasibility is based on the *semantical* notion of specificity, which makes use of the well-known concept of SLD-derivation from the field of logic programming (see Section 2.2). The characterizations with activation and path sets implement this clean semantic notion.

Finally, it must be remarked the comparison of arguments is embedded into a dialectical process, where arguments may be defeated, and there may exist defeaters for the defeaters, and so on. In DeLP a complete dialectical analysis is performed constructing a tree of arguments. Since this is not the subject of this paper, we refer the interested reader to [GAR 98, SIM 92] for details on the dialectical process. The syntactic criterion ($\geq$) for specificity can be used directly by the defeater notion (see Definition 2.9). Thus, the new definition for specificity can be embedded naturally in DeLP in a modular way.

4. Related Work

Next we will relate our work to other approaches to argumentation. In Section 4.1 we will relate our approach to other argumentation formalisms and their comparison criteria for conflicting arguments or default rules. We will then briefly discuss other frameworks for defeasible and default reasoning in Sections 4.2 and 4.3.

A more detailed overview and comparison of logical models of arguments is given in the survey article [CHE 00]. The journal article [CHE 02] relates the defeasible logic programming framework with specificity and its semantics to classical logic programming frameworks. It shows that the DeLP semantics is closely related to the well-founded semantics [GEL 88b] and the stable model semantics [GEL 88a] for normal logic programs. For more details the interested reader is referred to [CHE 02].

4.1. *Argumentation*

Dung and Son in [DUN 96] introduce an argumentation-theoretic approach to default reasoning with specificity. Default reasoning in general, and argumentative reasoning in particular, is defined in terms of a set E of evidence (or facts), and a pair $K = (D, B)$ which represents the *domain knowledge* consisting of a set of default rules D, and a first-order theory B representing background knowledge (Δ and Π_G in our notation). As stated before, our approach also takes into consideration the background knowledge B that was assumed empty in [DUN 96]. It is certainly interesting to consider a generalized setting, where evidence and background knowledge are not restricted to facts and strict rules, respectively. But this is beyond the scope of this paper.

In [DUN 96], the authors claim that most priority-based approaches define the semantics of Th wrt. certain partial orders on D, determined only by K. Let PO_K be the set of all partial orders defined in this way. For every partial order $\alpha \in PO_K$ (where $(d, d') \in \alpha$ means that d has lower priority than d'), we define $<_\alpha$ to be a partial order between sets of defaults in D, where $S <_\alpha S'$ means that S is preferred to S'. Whatever the definition of $<_\alpha$, it has to satisfy the following property: *Let S be a subset of D, and let d, d' be two defaults in D such that $(d, d') \in \alpha$. Then $S \cup \{d'\} <_\alpha S \cup \{d\}$.*

The partial order $<_\alpha$ can be extended into a partial order between models in $B \cup E$, by defining $M <_\alpha M'$ iff $D_M <_\alpha D_{M'}$, where D_M is the set of all defaults in D which are satisfiable in M. A default p/q is satisfiable in M iff the implication $p \rightarrow q$ is satisfiable in M. A model M of $B \cup E$ is a preferred model of Th iff there exists a partial order α in PO_K such that M is minimal wrt. $<_\alpha$. [DUN 96] shows that any preferential semantics based on $<_\alpha$ is not satisfactory enough since the set of evidence E is not considered.

Example 4.1 (Taken from [DUN 96]) *Consider the default theory $Th = (E, K)$, where $B = \emptyset$, D = {d/c, c/b, $d/\neg a$, b/a }, and E = $\{d\}$. The desirable semantics here is represented by the model M = $\{d, c, b, \neg a\}$. To have this semantics, most priority-based approaches assign the default b/a a lower priority than the default $d/\neg a$. Let us consider Th under a new set E = $\{d, \neg c, b\}$. Since c does not hold, the default $d/\neg a$ cannot be considered more specifi c than the default b/a, so that it should not be the case that either a or $\neg a$ are concluded.*

However, in any priority-based approach using the same priorities between defaults wrt. E and E', we have $M = \{\neg a, d, \neg c, b\} <_\alpha M' = \{a, d, \neg c, b\}$ since $D_M = \{c/b, d/\neg a\} <_\alpha D'_M = \{c/b, b/a\}$ (due to the fact that $(b/a, d/\neg a) \in \alpha$). Hence priority-based approaches would conclude $\neg a$ given (E', K), which is not the intuitive result, leading to the idea that default b/a should have a lower priority than $d/\neg a$ under evidence E, but a different priority under evidence E'.

Example 4.1 can be recast into the DeLP formalism by rewriting a default rule a/b as a defeasible clause $b \prec a$. Let us consider the preferred model associated with a DeLP program as defined by those literals supported by arguments ultimately undefeated. It turns out that the intuitively preferred model is computed correctly, since the evidence E is taken into account.

Example 4.2 *Consider the set* Δ = { $(a \prec b)$, $(\sim a \prec d)$, $(c \prec d)$, $(b \prec c)$} *of defeasible clauses, and let* $\Pi = \{d\}$. *In this case, we have arguments for* b, c, d, a *and* $\sim a$. *The argument for* $\sim a$ *is more specific than the argument for* a. *However, if* $\Pi = \{d\ , \sim c\ , b\}$, *we will have still undefeated arguments for* d, $\sim c$ *and* b, *but* $\sim a$ *will no longer hold (since it is blocked by the argument* $\{a \prec b\}$*).*

The previous example shows that in our approach, preference among defaults (defeasible rules) is determined dynamically during the dialectical analysis. This implies that our approach is context-sensitive as defined in [DUN 96] (although this is denied in the same reference). A distinctive feature of specificity is that it can be generalized to other common-sense reasoning approaches where the notion of *derivation* plays a central role. Thus specificity results as a useful comparison criterion for choosing between conflicting extensions in proof-theoretic approaches, whereas the dialectical analysis determines whether a given extension (argument) is ultimately preferred.

In [BEN 97], Benferhat and Garcia investigate a local approach to deal with conflicts in the presence of default rules. They suggest that when a conflict appears, the set of pieces of information that are responsible of this conflict are to be identified, and then (using a new definition of specificity) priorities should be attached to default rules inside each conflict. They claim that the resulting approach is modular, in the sense that the step of computing the specificity ordering of the defaults is independent of the step of solving conflicts. Hence, if another definition of specificity is preferred, then it is not very hard to adapt it to their approach. Note that this way of handling specificity differs from ours, in the sense that default rules are labeled with priorities inside each conflict, whereas our characterization defines preference just in terms of activation sets and derivation trees, without any particular priority relationship among defeasible rules. It must be remarked that both our approach and Benferhat and Garcia's [BEN 97] rely on the notion of *defeasible inference*. In [BEN 97] defeasible inheritance networks are used and depicted as trees or graphs; in our approach, defeasible inference is expressed in terms of arguments and dialectical trees. However, the underlying logical languages in both approaches differ. Benferhat and Garcia depart from a propositional logical language L, which differs from the language for defeasible logic programs (e. g., in [BEN 97] disjunctions in heads of rules are allowed).

In [AMG 96], Amgoud, Cayrol and Le Berre investigate the problem of defining preference relations to compare conflicting arguments. They state that two kinds of preference relations are most commonly encountered: *implicit relations*, which are

syntactically extracted from the belief base (in line with the specificity criterion presented in this paper), and *explicit relations* which are most often induced by a priority ordering on the belief base itself. Unlike our approach they focus on explicit preference relations. They discuss three explicit preference relations induced by a preference relation defined on the support of the arguments. The first preference criterion is defined in the context of possibilistic logic [BEN 93], and assumes a stratified belief base. The second and the third criteria (based on [ELV 93] and [CAY 93], respectively) assume that a partial pre-ordering $\leq$ is defined on the belief base.

The authors show that the first and the second approaches lead to contradictory results in some examples, and the third is a refinement of the first. Finally, they propose to apply their particular implementation of an Assumption-based Truth Maintenance System (ATMS) to argument-based reasoning taking into account a stratified base of clauses. It should be noted that the three relations proposed in this paper are based on explicit priorities between rules. In contrast, our approach focuses on comparing arguments, which involves a *set* of defeasible rules (Definition 2.7) without considering priorities between two rules.

Other argumentation formalisms—particularly those motivated by legal reasoning, such as [PRA 97]—consider priorities as well as defeasible reasoning about priorities. It must be remarked that in these cases criteria for comparing arguments are also debatable, and in many cases they are subordinated to hierarchical and temporal considerations (see [PRA 97] for an in-depth discussion). In contrast to these approaches, we concentrate on first finding an acceptable criterion for determining preferred extensions associated with the presence of defeasible information. Incorporating other features (such as hierarchical or temporal preference principles) is intended for further research.

4.2. *Prioritized Default Logic and Inheritance Reasoning*

Brewka and Eiter [BRE 00] have extended default logic in order to handle priorities, developing a *Prioritized Default Logic* (PDL). This approach has many properties which seem relevant for argumentation, such as explicit representation of preferences and reasoning about preferences. Although this approach is not explicitly argument-based, prioritized default theories extend default theories adding a strict partial order on defaults, using this ordering to define preferred extensions.

A prioritized default theory $\Delta = (D, W, <)$ extends the default theory (D, W) with a strict partial order $<$ on default rules. A default d will be considered preferred over default d' whenever $d < d'$ holds. Δ is called *fully prioritized* iff $<$ is a well-ordering. The following proposition can be established: if $\Delta = (D, W, <)$ is a fully prioritized ground theory, and E a classical extension of Δ, then E is a preferred extension of Δ iff for each default $d \in D$ such that $pre(d) \in E$ and $cons(d) \notin E$ there exists a set of defaults $K_d \subseteq \{d' \in GD(D, E) \mid d' < d\}$ such that d is defeated in $Th(W \cup cons(K_d))$. Given a default $d = a : b_1, \ldots, b_n/c$, where $a, b_1, \ldots, b_n, c$

are first-order formulas, a is called the *prerequisite* of d, each b_i is a *justification*, and c is the *consequent*. This is denoted as $pre(d)$, $jus(d)$ and $cons(d)$, respectively. Here $GD(D, E)$ denotes the set of all defaults from D which are generating in E (a default d is called generating in a set of well-formed formulae S if $pre(d) \in S$ and $\neg just(d) \cap S = \emptyset$. This proposition basically says that in preferred extensions defaults which are not applied must be defeated by defaults with higher priority.

PDL has a number of properties which seem to be relevant for defeasible argumentation, such as non-monotonicity, explicit representation of preferences and reasoning about preferences. In [BRE 00], it is proven that PDL satisfies two reasonable principles for preference handling, which distinguishes PDL from other approaches. However, since an ordering of defaults is enforced, similar problems to those mentioned in Section 4.1 are also present. Further, the approach in [BRE 00] considers sets of default rules only, not also strict rules, as done here. In addition, no procedures for prioritized default theories are investigated in [BRE 00].

In [GEL 90b], a formalization of inheritance reasoning in autoepistemic logic is presented. In this context, an autoepistemic theory Th is given by a set of propositional formulae augmented by a belief operator. Also strict rules besides defeasible rules are taken into consideration (as done here). However, in inheritance networks, rule bodies are restricted to containing at most one literal only—in contrast to the approach presented here.

The proposed formalization provides a completely axiomatic view on inheritance reasoning, introducing the notions belief sets and explanations. An explanation of an autoepistemic system is a set D of sentences such that the theory $Th \cup D$ has a stable expansion (i. e., there is a set E such that E is identical with the set of consequences from $Th \cup D \cup E$). In this approach, explanations are ordered by a pre-order $<$ (preferability relation).

The semantics is explained completely axiomatically in [GEL 90b], by introducing the notion of rank in an inheritance network, i. e. the length of the longest path which ends at a certain node. But the authors do not investigate operational procedures for inheritance networks. Rational principles are incorporated in the semantics: minimality and reliability of explanations. The semantics is similar to the one presented here, because minimal and more reliable explanations are preferred in both approaches.

4.3. *Logic Programming and Defeasible Logic with Superiority Relation*

In [KAK 94] and later in [DIM 95], *Logic Programming without Negation as Failure* (LPwNF) was introduced. A LPwNF program consists of a set of basic rules $L_0 \leftarrow L_1, \ldots, L_k$ (where L_i are literals that could be preceded by strong negation) and a given irreflexive and antisymmetric priority relation among program rules. The authors claim that default negation can be removed using the following transformation: the rule $r_0 = p \leftarrow q, not\ r$ is transformed into two rules, $r_1 = p \leftarrow q$ and $r_2 = \sim p \leftarrow r$, with $r_1 < r_2$. Hence, when r is not derivable the rule r_2 cannot be used,

and there is a derivation for p. On the other hand when r is derivable, rule r_2 blocks r_1. However, the problem with this approach is that when r is derivable, a new literal (not present in the original program) is derivable: $\sim p$. Contradiction between derivations is based on complementary literals, and the priority relation among rules. The proof procedure of LPwNF is very similar to the one of d-Prolog [COV 88, NUT 94].

Although in [DIM 95] there is no comparison with defeasible logic, in [ANT 00] a comparison among LPwNF, defeasible logic, and so-called courteous logic programs is given. The main result of [ANT 00] is that defeasible logic can prove everything that sceptical LPwNF can. In [GEL 97], Gelfond and Son developed a system to "investigate the methodology of reasoning with prioritized defaults in the language of logic programs under the answer set semantics". Their system allows the representation of defeasible and strict rules, and the representation of an order among those rules. The way in which defeasible inferences are obtained is very similar to [ANT 00], although no comparison of these two systems is given.

In [ANT 00, MAH 98], another approach for defeasible reasoning is presented. In this context, defeasible logic programs are (almost) identical to programs $\mathcal{P}$ as defined in Definition 2.3. But there, specificity is a relation between program clauses, modeled by the so-called *superiority* relation $>$, whereas in our framework specificity is an implicit relation between arguments according to Definition 3.1. The main difference is that this approach is not argument-based. Since the relation $>$ must be explicitly given by the programmer in addition to the program $\mathcal{P}$, we have to consider the pair $(\mathcal{P}, >)$ for this approach. Since the procedure for deriving defeasibly valid literals is quite different from our approach, it is not clear how to express specificity as defined here by means of an appropriately chosen superiority relation. However, the construction of such a relation is a non-trivial issue, and deserves a more detailed analysis.

5. Conclusions

Formalisms for representing common-sense knowledge need to deal with contradictory conclusions, and decide between them with some comparison criterion. To our opinion, this comparison should be performed within the formalism itself by analyzing the pieces of knowledge which lead to contradictory conclusions. Thus, our aim was to look forward for an autonomous comparison criterion that may fit in any rule-based formalism.

As a result we characterized a generalized version of specificity, based on the comparison criterion defined in [POO 85, SIM 92]. We showed that specificity can be redefined in terms of two different approaches: *activation sets* (Theorem 3.11) and *derivation trees* (Theorem 3.15). A syntactic criterion was obtained, which can be implemented in a computationally attractive way. This has been done in the DLP system (described in [GAR 97]) which implements the algorithm in Figure 3. These results may be applied to other rule-based formalisms which currently make use of explicit priorities.

Further work will concentrate on investigating even deeper the relationships to other approaches and possible translations from one method of defeasible reasoning into another one. For instance, it seems to be possible to reformulate defeasible reasoning as done here by means of (extended) logic programs (see also [DUN 96]). Last but not least, the integration of defeasible reasoning into agent programming should be tackled in greater detail.

Acknowledgements

This research has been supported by the German-Argentinian program on scientific and technological cooperation, funded by the *Bundesministerium für Bildung und Forschung* in Germany and the *Secretaría de Ciencia y Tecnología* in Argentina (see also [DIX 99]). A preliminary version of this paper appeared as [STO 00]. We thank some anonymous referees for a number of suggestions that helped to improve this article.

6. References

[ALF 96] ALFERES J. J., PEREIRA L. M., Eds., *Reasoning with Logic Programming*, LNAI 1111, Springer, Berlin, Heidelberg, New York, 1996.

[AMG 96] AMGOUD L., CAYROL C., BERRE D. L., "Comparing Arguments using Preference Orderings for Argument-based Reasoning", *Proc. of the 8th International Conference on Tools with Artificial Intelligence, ICTAI'96*, IEEE, 1996, p. 400-403.

[ANT 00] ANTONIOU G., MAHER M. J., BILLINGTON D., "Defeasible Logic versus Logic Programming without Negation as Failure", *Journal of Logic Programming*, vol. 42, 2000, p. 47-57.

[BEN 93] BENFERHAT S., DUBOIS D., PRADE H., "Argumentative Inference in Uncertain and Inconsistent Knowledge Bases", *Proc. of the 9th Conference on Uncertainty in AI*, 1993, p. 411-419.

[BEN 97] BENFERHAT S., GARCIA L., "A Coherence-Based Approach to Default Reasoning", GABBAY D. M., KRUSE R., NONNENGART A., OHLBACH H.-J., Eds., *Proceedings of 1st International Joint Conference on Qualitative and Quantitative Practical Reasoning*, LNAI 1244, Bad Honnef, 1997, Springer, Berlin, Heidelberg, New York, p. 43-57.

[BRE 00] BREWKA G., EITER T., "Prioritizing Default Logic", HÖLLDOBLER S., Ed., *Intellectics and Computational Logic: Papers in Honor of Wolfgang Bibel*, p. 27-46, Kluwer Academic Publishers, Dordrecht, Boston, London, 2000.

[CAY 93] CAYROL C., ROYER V., SUAREL C., "Managment of Preferences in Assumption-Based Reasoning", BOUCHON-MEUNIER B., VALVERDE L., YAGER R. R., Eds., *Proceedings of 4th International Conference on Processing and Management of Uncertainty in Knowledge-Based Systems 1992 – Advanced Methods in Artificial Intelligence*, LNCS 682, Springer, Berlin, Heidelberg, New York, 1993, p. 13-22.

[CHE 00] CHESÑEVAR C. I., MAGUITMAN A., LOUI R., "Logical Models of Argument", *ACM Computing Surveys*, vol. 32, num. 4, 2000, p. 337-383, ACM Press.

[CHE 02] CHESÑEVAR C. I., DIX J., STOLZENBURG F., SIMARI G. R., "Relating Defeasible and Normal Logic Programming through Transformation Properties", *Theoretical Computer Science*, , 2002, To appear.

[COV 88] COVINGTON M. A., NUTE D., VELLINO A., *Prolog Programming in Depth*, Scott, Foresman and Company, Glenview, IL, London, 1988.

[DIM 95] DIMOPOULOS Y., KAKAS A., "Logic Programming without Negation as Failure", *Proceedings of 5th. International Symposium on Logic Programming*, Cambridge, MA, 1995, MIT Press, p. 369–384.

[DIX 99] DIX J., STOLZENBURG F., SIMARI G. R., FILLOTTRANI P. R., "Automating Defeasible Reasoning with Logic Programming (DeReLoP)", JÄHNICHEN S., LOISEAU I., Eds., *Proceedings of the 2nd German-Argentinian Workshop on Information Technology*, Königswinter, 1999, p. 39-46.

[DUN 96] DUNG P. M., SON T. C., "An Argumentation-theoretic Approach to Reasoning with Specifi city", AIELLO L. C., DOYLE J., SHAPIRO S. C., Eds., *Proceedings of 5th International Conference on Principles of Knowledge Representation and Reasoning*, 1996, p. 506-517.

[ELV 93] ELVANG-GORANSONN M., FOX J., KRAUSE P., "Dialectic Rreasoning with Inconsistent Information", *Proc. of the 9th Conference on Uncertainty in AI*, 1993, p. 114-121.

[GAB 94] GABBAY D. M., HOGGER C. J., ROBINSON J. A., Eds., *Handbook of Logic in Artificial Intelligence and Logic Programming. Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning*, Oxford University Press, 1994.

[GAR 97] GARCÍA A. J., "Defeasible Logic Programming: Defi nition and Implementation", Master's thesis, Dep. de Ciencias de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina, Jul. 1997.

[GAR 98] GARCÍA A. J., SIMARI G. R., CHESÑEVAR C. I., "An Argumentative Framework for Reasoning with Inconsistent and Incomplete Information", *Workshop on Practical Reasoning and Rationality*, 13th biennial European Conference on Artifi cial Intelligence (ECAI-98), Aug. 1998.

[GAR 00] GARCÍA A. J., "Defeasible Logic Programming: Defi nition, Operational Semantics and Parallelism", PhD thesis, Computer Science Department, Universidad Nacional del Sur, Bahía Blanca, Argentina, Dec. 2000.

[GEL 88a] GELFOND M., LIFSCHITZ V., "The Stable Model Semantics for Logic Programming", KOWALSKI R., BOWEN K., Eds., *5th Conference on Logic Programming*, MIT Press, 1988, p. 1070-1080.

[GEL 88b] VAN GELDER A., ROSS K. A., SCHLIPF J. S., "Unfounded Sets and well-founded Semantics for general logic Programs", *Proceedings 7th Symposion on Principles of Database Systems*, 1988, p. 221-230.

[GEL 90a] GELFOND M., LIFSCHITZ V., "Logic Programs with Classical Negation", WARREN D., SZEREDI P., Eds., *Proceedings of the International Conference on Logic Programming*, MIT Press, 1990, p. 579-597.

[GEL 90b] GELFOND M., PRZYMUSINSKA H., "Formalization of Inheritance Reasoning in Autoepistemic Logic", *Fundamenta Informaticae*, vol. XIII, 1990, p. 403-443, IOS Press.

[GEL 97] GELFOND M., SON T. C., "Reasoning with Prioritized Defaults", *Selected Papers from the Workshop on Logic Programming and Knowledge Representation*, LNAI 1471, Springer, Berlin, Heidelberg, New York, 1997, p. 164–223.

[HOR 94] HORTY J. F., "Some Direct Theories of Nonmonotonic Inheritance", Gabbay et al. [GAB 94], p. 111-187.

[KAK 94] KAKAS A. C., MANCARELLA P., DUNG P. M., "The Acceptability Semantics for Logic Programs", *Proceedings of the 11th International Conference on Logic Programming*, Santa Margherita, Italy, 1994, MIT Press, p. 504–519.

[LIF 96] LIFSCHITZ V., "Foundations of Logic Programs", BREWKA G., Ed., *Principles of Knowledge Representation*, CSLI Publications, 1996.

[LLO 87] LLOYD J. W., *Foundations of Logic Programming*, Springer, Berlin, Heidelberg, New York, 1987.

[LOU 87] LOUI R. P., "Defeat Among Arguments: A System of Defeasible Inference", *Computational Intelligence*, vol. 3, num. 3, 1987, p. 100-106.

[MAH 98] MAHER M. J., ANTONIOU G., BILLINGTON D., "A Study of Provability in Defeasible Logic", SLANEY J., ANTONIOU G., Eds., *Proceedings of 11th Australian Joint Conference on Artificial Intelligence*, LNAI 1502, Springer, Berlin, Heidelberg, New York, 1998, p. 215-226.

[MUR 01] MURRAY J., OBST O., STOLZENBURG F., "Towards a Logical Approach for Soccer Agents Engineering", STONE P., BALCH T., KRAETZSCHMAR G., Eds., *RoboCup 2000: Robot Soccer World Cup IV*, LNAI 2019, p. 199-208, Springer, Berlin, Heidelberg, New York, 2001.

[NUT 94] NUTE D., "Defeasible Logic", Gabbay et al. [GAB 94], p. 355-395.

[POL 91] POLLOCK J. L., "Self-Defeating Arguments", *Minds and Machines*, vol. 1, num. 4, 1991, Special issue on *Defeasible Reasoning*.

[POO 85] POOLE D. L., "On the Comparison of Theories: Preferring the Most Specific Explanation", *Proceedings of 9th International Joint Conference on Artificial Intelligence*, IJCAI Inc., San Mateo, CA, Morgan Kaufmann, Los Altos, CA, 1985, p. 144-147.

[PRA 97] PRAKKEN H., SARTOR G., "Argument-based logic programming with defeasible priorities", *Journal of Applied Non-classical Logics*, vol. 7, 1997, p. 25-75.

[SIM 92] SIMARI G. R., LOUI R. P., "A Mathematical Treatment of Defeasible Reasoning and its Implementation", *Artificial Intelligence*, vol. 53, 1992, p. 125-157.

[SIM 94] SIMARI G. R., CHESÑEVAR C. I., GARCÍA A. J., "The Role of Dialectics in Defeasible Argumentation", *Anales de la XIV Conferencia Internacional de la Sociedad Chilena para Ciencias de la Computación*, Universidad de Concepción, Concepción (Chile), Nov. 1994.

[STO 00] STOLZENBURG F., GARCÍA A. J., CHESÑEVAR C. I., SIMARI G. R., "Introducing Generalized Specificity in Logic Programming", FEIERHERD G. E., Ed., *Proceedings of the 6th Argentine Congress on Computer Science*, Ushuaia, Argentina, 2000, JAIIO, Buenos Aires, p. 359-370.

[VRE 97] VREESWIJK G. A., "Abstract Argumentation Systems", *Artificial Intelligence*, vol. 90, 1997, p. 225-279.

[WAN 97] WANG X., YOU J., YUAN L., "Logic Programming without Default Negation Revisited", *Proceedings of IEEE International Conference on Intelligent Processing Systems*, IEEE, 1997, p. 1169–1174.