View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by University of Queensland eSpace

A Comparison of Sceptical NAF-Free Logic Programming Approaches

G. Antoniou, M.J. Maher, Billington, G. Governatori
CIT, Griffith University
Nathan, QLD 4111, Australia
{ga,mjm,db,guido}@cit.gu.edu.au

Abstract

Recently there has been increased interest in logic programming-based default reasoning approaches which are not using negation-as-failure in their object language. Instead, default reasoning is modelled by rules and a priority relation among them.

Historically the first logic in this class was Defeasible Logic. In this paper we will study its relationship to other approaches which also rely on the idea of using logic rules and priorities. In particular we will study sceptical LPwNF, courteous logic programs, and priority logic.

1 Introduction

Recently there has been increased interest in modelling default reasoning by means of rules *without negation as failure*, and a priority relation. In fact [16] argues that the concept of priority is more basic than the concept of a default.

Defeasible Logic [11, 12] is an early approach to sceptical nonmonotonic reasoning [1] which was based on rules without negation as failure, plus a priority relation. In fact it has an implementation as a straightforward extension of Prolog [5].

LPwNF (Logic Programming without Negation as Failure) is a recent approach, introduced in [6]. It supports both credulous and sceptical reasoning, unlike defeasible logic, and has an argumentation-theoretic characterisation.

The main contribution of this paper is to compare defeasible logic with sceptical LPwNF. We discuss how the two approaches differ. The main difference is that LPwNF does not take into account *teams of rules* [7] supporting a conclusion, but rather views rules individually. By doing so, LPwNF fails to draw desirable conclusions that defeasible logic can, as we show in this paper. On the other hand, defeasible logic can prove everything that sceptical LPwNF can.

Also we compare defeasible logic with courteous logic programs [7] and priority logic [16, 17]. Finally we point out at an earlier result which establishes a relationship between defeasible logic and inheritance networks.

M. Gelfond, N. Leone and G. Pfeifer (eds)
Logic Programming and Non-monotonic Reasoning. 5th International Conference, LPNMR'99
LNCS 1730, pp. 347–357, 1999.
© Springer 1999. The original publication is available at www.springerlink.com.

2 Defeasible Logic

In this paper we restrict attention to propositional defeasible logic, and assume that the reader is familiar with the notation and basic notions of propositional logic. If q is a literal, $\sim q$ denotes the complementary literal (if q is a positive literal p then $\sim q$ is $\neg p$; and if q is $\neg p$, then $\sim q$ is p).

A *rule* r consists of its *antecedent* $A(r)$ (written on the left; $A(r)$ may be omitted if it is the empty set) which is a finite set of literals, an arrow, and its *consequent* (or *head*) $C(r)$ which is a literal. In writing rules we omit set notation for antecedents. There are three kinds of rules: *Strict rules* are denoted by $A \rightarrow p$ and represent indisputable conclusions ("Emus are birds"); *defeasible rules* are denoted by $A \Rightarrow p$ and represent conclusions that can be defeated by contrary evidence ("Birds usually fly"); and defeaters are denoted by $A \rightsquigarrow p$ and represent knowledge which might prevent the conclusion $\neg p$ from being drawn without directly supporting the conclusion p ("Heavy animals may not fly"). Given a set R of rules, we denote the set of all strict rules in R by R_s, and the set of strict and defeasible rules in R by R_{sd}. $R[q]$ denotes the set of rules in R with consequent q.

In the following we use the formalization of [4]. A *superiority relation on* R is an acyclic relation $>$ on R (that is, the transitive closure of $>$ is irreflexive), and is used to represent priority information among rules. A *defeasible theory* T is a triple $(F, R, >)$ where F is a finite set of literals (called *facts*), R a finite set of rules, and $>$ a superiority relation on R.

A *conclusion* of T is a tagged literal and can have one of the following four forms:

- $+\Delta q$, which is intended to mean that q is definitely provable in T.
- $-\Delta q$, which is intended to mean that we have proved that q is not definitely provable in T.
- $+\partial q$, which is intended to mean that q is defeasibly provable in T.
- $-\partial q$ which is intended to mean that we have proved that q is not defeasibly provable in T.

A *derivation* (or *proof*) in $T = (F, R, >)$ is a finite sequence $P = (P(1), \ldots P(n))$ of tagged literals satisfying the following conditions ($P(1..i)$ denotes the initial part of the sequence P of length i):

$+\Delta$: If $P(i+1) = +\Delta q$ then either
$q \in F$ or
$\exists r \in R_s[q]\ \forall a \in A(r) : +\Delta a \in P(1..i)$

$-\Delta$: If $P(i+1) = -\Delta q$ then
$q \notin F$ and

$$\forall r \in R_s[q]\ \exists a \in A(r) : -\Delta a \in P(1..i)$$

$+\partial$: If $P(i+1) = +\partial q$ then either
(1) $+\Delta q \in P(1..i)$ or
(2) $\exists r \in R_{sd}[q]$ such that
(2.1) $\forall a \in A(r) : +\partial a \in P(1..i)$ and
(2.2) $-\Delta \sim q \in P(1..i)$ and
(2.3) $\forall s \in R[\sim q]$, either
(2.3.1) $\exists a \in A(s) : -\partial a \in P(1..i)$ or
(2.3.2) $\exists t \in R_{sd}[q]$ such that
$\forall a \in A(t) : +\partial a \in P(1..i)$ and $t > s$

$-\partial$: If $P(i+1) = -\partial q$ then
(1) $-\Delta q \in P(1..i)$ and
(2) (2.1) $\forall r \in R_{sd}[q]\ \exists a \in A(r) : -\partial a \in P(1..i)$ or
(2.2) $+\Delta \sim q \in P(1..i)$ or
(2.3) $\exists s \in R[\sim q]$ such that
(2.3.1) $\forall a \in A(s) : +\partial a \in P(1..i)$ and
(2.3.2) $\forall t \in R_{sd}[q]$ either
$\exists a \in A(t) : -\partial a \in P(1..i)$ or $t \not> s$

The elements of a derivation are called *lines* of the derivation. We say that a tagged literal L is *provable* (or *derivable*) in $T = (F, R, >)$, denoted $T \vdash L$, iff there is a derivation in T such that L is a line of a proof P.

Even though the definition seems complicated, it follows ideas which are intuitively appealing. For an explanation of this definition see [10].

In the remainder of this paper we will only need to consider defeasible rules and a superiority relation; facts, strict rules and defeaters will not be necessary.

3 LPwNF

In LPwNF [6], a logic program consist of a set of rules of the form $p \leftarrow q_1, \ldots, q_n$, where $p, q_1, \ldots, q_n$ are literals, and an irreflexive and transitive priority relation $>$ among rules.

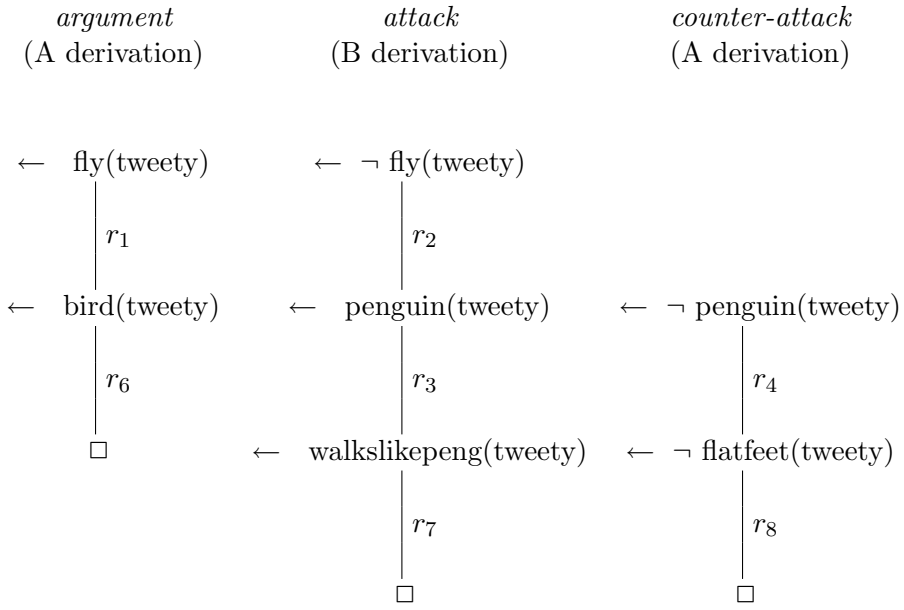
[6] introduced a proof theory and a corresponding argumentation framework. The main idea of LPwNF is the following: In order to prove a literal q, a *type A derivation* must be found which proves q. One part of this derivation is a top-level proof of q in the sense of logic programming (SLD-resolution). But additionally every attack on this argument must be counterattacked. Attacks are generated in *type B derivations*. For an A derivation to succeed all B derivations must fail.

In general, a rule r in a type B derivation can attack a rule r' in a type A derivation if they have complementary heads, and r is not weaker than r', that

is, $r' \not> r$. On the other hand, a rule r in a type A derivation can attack a rule r' in a type B derivation if they have complementary heads, and $r > r'$. This reflects the notion of scepticism: it should be easier to attack a positive argument than to counterattack (i.e. attack the attacker). For example, consider the following program from [6]:

$r_1 : \ fly(X) \leftarrow bird(X)$
$r_2 : \ \neg fly(X) \leftarrow penguin(X)$
$r_3 : \ penguin(X) \leftarrow walkslikepeng(X)$
$r_4 : \ \neg penguin(X) \leftarrow \neg flatfeet(X)$
$r_2 > r_1$
$r_4 > r_3$
$r_5 : \ bird(X) \leftarrow penguin(X)$
$r_6 : \ bird(tweety) \leftarrow$
$r_7 : \ walkslikepeng(tweety) \leftarrow$
$r_8 : \ \neg flatfeet(tweety) \leftarrow$

Here it is possible to prove $fly(tweety)$. Firstly there is a standard SLD refutation (A derivation) of $\leftarrow fly(tweety)$ via the rules r_1 and r_6. Additionally we need to consider all possible attacks on this refutation. In our case, r_1 can be attacked by r_2. Thus we start a B derivation with goal $\leftarrow \neg fly(tweety)$ (with first rule r_2), and have to show that this proof fails. This happens because the rule r_3 is successfully counterattacked by r_4. There are no other attacks on the original derivation. The following figure illustrates how the reasoning proceeds.



Below we give the formal definition. LPwNF can support either credulous or sceptical reasoning. Since in this paper we are interested in a comparison with defeasible logic, we will restrict ourselves to the sceptical case (as we have already done so far in this section). Also, our presentation is slightly simpler than that of [6]. The reason is that in their paper, Dimopoulos and Kakas showed the soundness

of their proof theory w.r.t. an argumentation framework, and they had to make the definition of derivations more complicated to collect the appropriate rules which are used to build an appropriate argument. This is not our concern here, so we just focus on the derivation of formulae.

A *type A derivation* from (G_1, r) to $G_n, r)$ is a sequence $((G_1, r), (G_2, r), \ldots, (G_n, r)$, where r is a rule, and each G_i has the form $\leftarrow q, Q$, where q is the selected literal and Q a sequence of literals. For G_i, $i \geq 1$, if there is a rule r_i such that either

1. $i = 1$, $r_i > r$, r_i resolves with G_i on q, and there is a type B derivation from $(\{\leftarrow\sim q\}, r_i)$ to $(\emptyset, r_i)$, or
2. $i > 1$, r_i resolves with G_i on q, and there is a type B derivation from $(\{\leftarrow\sim q\}, r_i)$ to $(\emptyset, r_i)$

then G_{i+1} is the resolvent of r_i with G_i.

A *type B derivation* from (F_1, r) to (F_n, r) is a sequence $(F_1, r), (F_2, r), \ldots, (F_n, r)$, where every F_i is of the form $F_i = \{\leftarrow q, Q\} \cup F_i'$, q the selected literal, and F_{i+1} is constructed from F_i as follows:

1. For $i = 1$, F_1 must have the form $\leftarrow q$. Let R be the set of rules r_i which resolve with $\leftarrow q$, and which satisfy the condition $r_i \not< r$. Let C be the set of resolvents of $\leftarrow q$ with the rules in R. If $[] \notin C$ then $F_2 = C$; otherwise there is no F_2.
2. For $i > 1$, let R be the set of rules r_i which resolve with $\leftarrow q, Q$ on q. Let R' be the subset of R containing all rules r_i such that there is no A derivation from $(\leftarrow\sim q, r_i)$ to $([], r_i)$. Let C be the set of all resolvents of the rules in R' with the rule $\leftarrow q, Q$, by resolving on q. If $[] \notin C$ then $F_{i+1} = C \cup F_i'$; otherwise there is no F_{i+1}.

4 A Comparison of LPwNF and Defeasible Logic

Given a logic program without negation as failure P, let $T(P)$ be the defeasible theory containing the same rules as P, written as defeasible rules, and the same superiority relation. In other words, rules in LPwNF are represented as defeasible rules in defeasible logic.

First we show that every conclusion provable in LPwNF can be derived in defeasible logic. The proof goes by induction on the length of a derivation and is found in the full version of this paper.

Theorem 4.1 *Let q be a literal which can be sceptically proven in the logic program without negation as failure P, that is, there is a type A derivation from $(\leftarrow q, r)$ to $([], r)$ for some rule r. Then $T(P) \vdash +\partial q$.*

However the reverse is not true. The reason is that *LPwNF argues on the basis of individual rules*, whereas *defeasible logic argues on the basis of teams of rules with the same head*. The difference can be illustrated by the following simple example.

$r_1 : \ monotreme(X) \Rightarrow mammal(X)$ $\quad$ $monotreme(platypus)$
$r_2 : \ hasFur(X) \Rightarrow mammal(X)$ $\quad$ $hasFur(platypus)$
$r_3 : \ laysEggs(X) \Rightarrow \neg mammal(X)$ $\quad$ $laysEggs(platypus)$
$r_4 : \ hasBill(X) \Rightarrow \neg mammal(X)$ $\quad$ $hasBill(platypus)$
$r_1 > r_3$
$r_2 > r_4$

Intuitively we conclude that *platypus* is a *mammal* because for every reason against this conclusion (r_3 and r_4) there is a stronger reason for $mammal(platypus)$ (r_1 and r_2 respectively). It is easy to see that $+\partial mammal(platypus)$ is indeed provable in defeasible logic: there is a rule in support of $mammal(platypus)$, and every rule for $\neg mammal(platypus)$ is overridden by a rule for $mammal(platypus)$.

On the other hand, the corresponding logic program without negation as failure is unable to prove $mammal(platypus)$: If we start with r_1, trying to build an A derivation, then we must counter the attack r_4 (which is not inferior to r_1) used in a B derivation. But LPwNF does not allow counterattacks on r_4 by another rule with head $mammal(platypus)$, but only by an attack on the body of r_4. The latter is impossible in our case (there is no rule matching $\neg hasBill(platypus)$). Thus the attack via r_4 succeeds and the proof of $mammal(platypus)$ via r_1 fails. Similarly, the proof of $mammal(platypus)$ via r_2 fails, due to an attack via rule r_3. Thus $mammal(platypus)$ cannot be proven.

It is instructive that even if LPwNF is modified to allow counterattacks on the same literal on which a rule r attacks a type A derivation, still we would not get the desired conclusion in the example above. With this modification, r_1 is attacked by r_4, which is counterattacked by r_2, which is attacked by r_3, which is counterattacked by r_1, which is attacked by r_4 etc. Defeasible logic breaks this cycle by recognising that any rule attacking the argument can be "trumped" by a superior rule supporting the argument. This difference illustrates once again the absence of the idea of a team of rules in LPwNF.

Our analysis so far has shown that defeasible logic is stronger than LPwNF because it allows attacks to be counterattacked by different rules. But note that a counterattacking rule needs to be stronger than the attacking rule. Thus it is not surprising that if the priority relation is empty, both approaches coincide.

Theorem 4.2 *Let P be a logic program without negation as failure with empty priority relation. Then a literal q can be sceptically proven in P iff $T(P) \vdash +\partial q$.*

5 Other Approaches

5.1 Courteous Logic Programs

Courteous logic programs [7] share some basic ideas of defeasible logic. In particular, the approach is logic programming based, implements sceptical reasoning, and is based on competing teams of rules, and a priority relation. It imposes a total stratification on the logic program by demanding that the atom dependency graph be acyclic. This ensures that each stratum contains only rules with head p or $\neg p$. An *answer set* is built gradually, stratum by stratum.

Compared to defeasible logic, courteous logic programs are more specialized in the following respects: (i) The atom dependency graph of a courteous logic program must be acyclic. This condition is central in the courteous logic program framework, but is not necessary in defeasible logic; (ii) Defeasible logic distinguishes between strict and defeasible conclusions, courteous logic programs do not. Thus defeasible logic is more fine-grained; (iii) Defeasible logic has the concept of a defeater, courteous logic programs do not. Thus defeasible logic offers a greater flexibility in the expression of information.

On the other hand, there seems to be a major difference between the two approaches, in that courteous logic programs may use negation as failure. However, a courteous logic program with negation as failure C can be modularly translated into a program C' without negation as failure: Every rule

$$r: \ L \leftarrow L1 \wedge \ldots \wedge L_n \wedge \ fail \ M_1 \wedge \ldots \wedge \ fail \ M_k$$

can be replaced by the rules:

$$r: \ L \leftarrow L_1 \wedge \ldots \wedge L_n \wedge p_r$$
$$p_r \leftarrow$$
$$\neg p_r \leftarrow M_1$$
$$\ldots$$
$$\neg p_r \leftarrow M_k$$

where p_r is a new propositional atom. If we restrict attention to the language of C, the programs C and C' have the same answer set.

Thus, without loss of generality we may assume that a courteous logic program C does not use negation as failure. The corresponding defeasible theory $df(C)$ is obtained by representing every rule in C' by an equivalent defeasible rule, and by using the same priority relation as C.

Theorem 5.1 *Let C be a courteous logic program. A literal q is in the answer set of C iff $df(C) \vdash +\partial q$.*

5.2 Priority Logic

Priority logic [16, 17] is a knowledge representation language where a theory consists of logic programming-like rules, and a priority relation among them. The meaning of the priority relation is that once a rule r is included in an argument, all rules inferior to r are automatically blocked from being included in the same argument. The semantics of priority logic is based on the notion of a *stable argument* for the credulous case, and the *well-founded argument* for the sceptical case.

Priority logic is a general framework with many instantiations (based on so-called *extensibility functions*), and supports both credulous and sceptical reasoning. To allow a fair comparison to defeasible logic, one has to impose the following restrictions: (i) We will only consider defeasible rules in the sense of defeasible logic. That is, we will not distinguish between strict and defeasible rules, and we will restrict attention to rules in which only propositional literals occur (but not more general formulae, as in priority logic). Also, there will be no defeaters. (ii) The priority/superiority relation will only be defined on pairs of rules with complementary heads. (iii) We will consider the two basic instantiations of priority logic, as determined by the extensibility functions $\mathcal{R}_1$ and $\mathcal{R}_2$ (see [16, 17] for details). (iv) We will compare defeasible logic to the sceptical interpretation of priority logic.

Under these conditions, the difference between defeasible logic and priority logic is highlighted by the following example:

$r_1 : \ quaker \leftarrow$ $\qquad$ $r_5 : \ footballfan \leftarrow republican$
$r_2 : \ republican \leftarrow$ $\qquad$ $r_6 : \ antimilitary \leftarrow pacifist$
$r_3 : \ pacifist \leftarrow quaker$ $\qquad$ $r_7 : \ \neg antimilitary \leftarrow footballfan$
$r_4 : \ \neg pacifist \leftarrow republican$
The priority relation is empty.

(Obviously in defeasible logic we consider r_1-r_7 to be defeasible rules.) In priority logic, if we use the extensibility relation $\mathcal{R}_1$, then the well-founded argument is the set of all rules, and therefore inconsistent. On the other hand, in the defeasible logic version T of the priority logic program, $T \not\vdash +\partial pacifist$, so the approaches are different.

And if we use the extensibility relation $\mathcal{R}_2$, then priority logic does not allow one to prove $\neg antimilitary$. But defeasible logic can prove $+\partial\neg antimilitary$. The difference is caused by the fact that defeasible logic does not propagate ambiguity, as extension-based formalisms like priority logic do (for a discussion of this issue see [14]).

5.3 Inheritance Networks

Nonmonotonic inheritance networks [13, 9] were an early nonmonotonic reasoning approach which had powerful implementations, even though they lacked declarativity. Moreover they are based on the use of rules and an implicit notion of priority

among rules. In [3] it was shown that inheritance networks as defined in [8] can be represented in defeasible logic. We outline the translation below.

A nonmonotonic inheritance network consists of a set of objects, a set of properties, and a set of arcs which is acyclic. Below is a list of the possible kinds of arcs, where a is an object, and p and q are properties (we use a variation of syntax to be consistent with this paper):

> $a \Rightarrow p$, meaning that a has the property p.
>
> $a \not\Rightarrow p$, meaning that a does not have property p.
>
> $p \Rightarrow q$, meaning that an object with property p typically has property q.
>
> $p \not\Rightarrow q$, meaning that an object with property p typically does not have property q.

A nonmonotonic inheritance network N is naturally translated into a defeasible theory $T(N)$:

> For every arc $a \Rightarrow p$ in N include the fact $p(a)$ in $T(N)$.
>
> For every $a \not\Rightarrow p$ in N include the fact $\neg p(a)$ in $T(N)$.
>
> For every path $a \Rightarrow \ldots \Rightarrow p \Rightarrow q$ in N include the rule $p(a) \Rightarrow q(a)$ in $T(N)$.
>
> For every path $a \Rightarrow \ldots \Rightarrow p \not\Rightarrow q$ in N include the rule $p(a) \Rightarrow \neg q(a)$ in $T(N)$.

We have omitted the definition of the superiority relation which simulates specificity in the inheritance networks of [8]. The complicated definition is found in [3]. That paper also proposes a way of compiling specificity into the definition of a derivation, which can be used to make the translation of a nonmonotonic inheritance network into a defeasible theory modular.

Result 5.2 *Let N be a nonmonotonic inheritance network. Then we may construct a defeasible theory $T(N)$, such that for every literal q, q is supported by N iff $T(N) \vdash +\partial q$.*

6 Conclusion

We have looked at the relationship between four logic programming-based formalisms that employ a priority relation among rules and take a sceptical approach to inference. Three, defeasible logic, LPwNF and courteous logic programs, belong to the same "school" of conservative reasoning in the classification of [15], while priority logic takes a fundamentally different approach, which is evident in its propagation of ambiguity. In addition, we showed that a class of nonmonotonic

inheritance networks can be simulated by defeasible logic, so it belongs, too, to the school of conservative reasoning, even though it is not a logical formalism.

Of the four formalisms in the conservative reasoning school, defeasible logic is the most powerful. It is able to draw more conclusions (from the same rules) than LPwNF can, principally because it argues on the basis of teams of rules. Courteous logic programs also employ teams of rules, but the approach is severely restricted in that the atom dependency graph is required to be acyclic. In addition, of course, defeasible logic makes a distinction between definite knowledge (obtained by facts and strict rules) and defeasible knowledge, and admits the use of defeaters.

The results of this paper indicate that defeasible logic deserves more attention. In other papers [2, 10] we have studied the logic as a formal system, including representation results, properties of the inference relation, and semantics.

References

[1] G. Antoniou. *Nonmonotonic Reasoning*. MIT Press 1997.

[2] G. Antoniou, D. Billington, and M.J. Maher. Normal forms for defeasible logic. In *Proc. 1998 Joint International Conference and Symposium on Logic Programming*, MIT Press 1998.

[3] D. Billington, K. de Coster and D. Nute. A modular translation from defeasible nets to defeasible logic. *Journal of Experimental and Theoretical Artificial Intelligence* 2 (1990): 151-177.

[4] D. Billington. Defeasible Logic is Stable. *Journal of Logic and Computation* 3 (1993): 370–400.

[5] M.A. Covington, D. Nute and A. Vellino. *Prolog Programming in Depth*. Prentice Hall 1997.

[6] Y. Dimopoulos and A. Kakas. Logic Programming without Negation as Failure. In *Proc. ICLP-95*, MIT Press 1995.

[7] B.N. Grosof. Prioritized Conflict Handling for Logic Programs. In *Proc. Int. Logic Programming Symposium*, J. Maluszynski (Ed.), 197–211. MIT Press, 1997.

[8] J.F. Horty, R.H. Thomason and D. Touretzky. A skeptical theory of inheritance in nonmonotonic semantic networks. In *Proc. AAAI-87*, 358-363.

[9] J.F. Horty. Some direct theories of nonmonotonic inheritance. In D.M. Gabbay, C.J. Hogger and J.A. Robinson (eds): *Handbook of Logic in Artificial Intelligence and Logic Programming* Vol. 3, Clarendon Press 1994, 111-187.

[10] M.J. Maher, G. Antoniou and D. Billington. A Study of Provability in Defeasible Logic. In *Proc. 11th Australian Joint Conference on Artificial Intelligence*, LNAI 1502, Springer 1998, 215-226.

[11] D. Nute. Defeasible Reasoning. In *Proc. 20th Hawaii International Conference on Systems Science*, IEEE Press 1987, 470–477.

[12] D. Nute. Defeasible Logic. In D.M. Gabbay, C.J. Hogger and J.A. Robinson (eds.): *Handbook of Logic in Artificial Intelligence and Logic Programming Vol. 3*, Oxford University Press 1994, 353-395.

[13] D. Touretzky. *The mathematics of inheritance systems*. Morgan Kaufmann 1986.

[14] D. Touretzky, J.F. Horty and R.H. Thomason. A clash of intuitions: The current state of nonmonotonic multiple inheritance systems. In *Proc. IJCAI-87*, 476-482, Morgan Kaufmann 1987.

[15] G. Wagner. Ex contradictione nihil sequitur. In *Proc. 12th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann 1991.

[16] X. Wang, J. You and L. Yuan. Nonmonotonic reasoning by monotonic inferences with priority constraints. In *Nonmonotonic Extensions of Logic Programming*, J. Dix, P. Pereira, and T. Przymusinski (eds), LNAI 1216, Springer 1997, 91-109.

[17] X. Wang, J. You and L. Yuan. Logic programming without default negation revisited. In *Proc. IEEE International Conference on Intelligent Processing Systems*, IEEE 1997.