Defeasible logic versus Logic Programming without Negation as Failure

G. Antoniou*, M.J. Maher, D. Billington
School of Computing & Information Technology, Griffith University, Nathan, QLD 4111, Australia

Received 3 November 1998; received in revised form 4 June 1999; accepted 4 June 1999

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. In this paper we compare the expressive power of two approaches in this family of logics: Defeasible Logic, and sceptical Logic Programming without Negation as Failure (LPwNF). Our results show that the former has a strictly stronger expressive power. The difference is caused by the latter logic’s failure to capture the idea of teams of rules supporting a specific conclusion. © 2000 Published by Elsevier Science Inc. All rights reserved.

Keywords: Defeasible logic; Logic programming

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 [12] argues that the concept of priority is more basic than the concept of a default.

Defeasible Logic [9,10] 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 [4].

Logic Programming without Negation as Failure (LPwNF) is a recent approach, introduced in [7] and studied in [5]. It supports both credulous and sceptical reasoning, unlike defeasible logic, and has an argumentation-theoretic characterisation.

The 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 [6] supporting a conclusion (all rules with the same head form a team), 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.

Finally we briefly compare defeasible logic with two further approaches, courteous logic programs [6] and priority logic [12,13].

2. Basics of 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 \( A(r) \rightarrow p \) and represent indisputable conclusions ("Emus are birds"); defeasible rules are denoted by \( A(r) \rightarrow \neg p \) and represent conclusions that can be defeated by contrary evidence ("Birds usually fly"); and defeaters are denoted by \( A(r) \rightarrow \neg p \) and represent knowledge which might prevent the conclusion \( p \) 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 \( R_{sd} \).

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
  \[
  q \in F \quad \text{or} \quad \exists r \in R_s[q] \ \forall a \in A(r) : +\Delta a \in P(1..i)
  \]

- \( -\Delta \): If \( P(i+1) = -\Delta q \) then
  \[
  q \notin F \quad \text{and} \quad \forall r \in R_s[q] \ \exists a \in A(r) : -\Delta a \in P(1..i)
  \]

- \( +\partial \): If \( P(i+1) = +\partial q \) then either
  \[
  (1) +\Delta q \in P(1..i) \quad \text{or}
  \]

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

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

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 [8].

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.

**Example 1** (adapted from [5]).

\[
\begin{align*}
r_1 : & \text{bird}(X) \Rightarrow \text{fly}(X) \quad r_5 : \text{penguin}(X) \Rightarrow \text{bird}(X) \\
r_2 : & \text{penguin}(X) \Rightarrow \neg \text{fly}(X) \quad f_1 : \text{bird}('\text{tweety}') \\
r_3 : & \text{walkslikepeng}(X) \Rightarrow \text{penguin}(X) \quad f_2 : \text{walkslikepeng}('\text{tweety}') \\
r_4 : & \neg \text{flatfeet}(X) \Rightarrow \neg \text{penguin}(X) \quad f_3 : \neg \text{flatfeet}('\text{tweety}')
\end{align*}
\]

\[
\begin{align*}
r_2 > r_1 \\
r_4 > r_3 
\end{align*}
\]

We can derive \(+\partial \neg \text{penguin('tweety')}\) because both rules \( r_3 \) and \( r_4 \) are applicable (with instantiation 'tweety') and \( r_4 \) is stronger than \( r_3 \). For the same reason we can derive \( -\partial \text{penguin('tweety')}\).

The fact \( f_1 \) allows us to derive \(+\Delta \text{bird('tweety')}\), thus also \(+\partial \text{bird('tweety')}\). Therefore rule \( r_1 \) (with instantiation 'tweety') is applicable. Moreover rule \( r_2 \), the only possible way for proving \( \neg \text{fly('tweety')}\), cannot be applied because we have already derived \( -\partial \text{penguin('tweety')}\). Thus we can derive \( \text{fly('tweety')}\).

3. LPwNF

In LPwNF [5], a logic program consists 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 antisymmetric priority relation \( > \) among rules.
Ref. [5] 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, $r' \nless r$. 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 which is the same as Example 1, but for variations of syntax.

Example 2.

Here it is possible to prove $fly(tweety)$. Firstly there is a standard SLD refutation ($A$ derivation) of $fly(tweety)$ 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 $\neg fly(tweety)$ (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 [5]. 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 \(\langle \neg q, Q \rangle\), 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

1. \(i = 1, r_i > r, r_i\) resolves with \(G_i\) on \(q\), and there is a type B derivation from \((\langle \neg \neg q \rangle, r_i)\) to \((\emptyset, r_i)\) or

2. \(i > 1, r_i\) resolves with \(G_i\) on \(q\), and there is a type B derivation from \((\langle \neg \neg q \rangle, r_i)\) 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 \(F_i = \langle \neg q, Q \rangle \cup F'_i\), \(q\) the selected literal, and \(F_{i+1}\) is constructed from \(F_i\) as follows:

1. For \(i = 1, F_1\) must have the form \(\neg q\). Let \(R\) be the set of rules \(r_i\) which resolve with \(\neg q\), and which satisfy the condition \(r_i \not\in R\). Let \(C\) be the set of resolvents of \(\neg q\) with the rules in \(R\). If \(\emptyset \not\in C\) 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 \(\neg q, Q\) on \(q\). Let \(R'\) be the subset of \(R\) containing all rules \(r_i\) such that there is no A derivation from \((\neg \neg q, r_i)\) to \((\emptyset, r_i)\). Let \(C\) be the set of all resolvents of the rules in \(R'\) with the rule \(\neg q, Q\), by resolving on \(q\). If \(\emptyset \not\in C\) 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.

If we study the used in the previous sections, we see that in both approaches we are able to prove \(\text{fly(tweety)}\). Moreover the idea used in both approaches is similar: Firstly we need a reasoning chain that leads to this conclusion, the application of rule \(r_1\) being the final step. And secondly we need to consider a possible attack by rule \(r_2\), and need to make sure that the rule does not apply because we cannot prove \(\text{penguin(tweety)}\).

A difference is that LPwNF takes a top down approach while a derivation in defeasible logic works bottom up. But this difference is not fundamental, and we are able to show that every conclusion provable in LPwNF can be derived in defeasible logic.
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 \( (\emptyset), r \) for some rule \( r \). Then \( T(P) \vdash +\partial q \).

We need the following technical result from [2].

Lemma 4.1. Let \( T \) be a defeasible theory containing no strict rules. If \( T \vdash +\partial q \) then \( T \vdash -\partial \sim q \).

The theorem follows directly from the following result.

Lemma 4.2. Suppose that there is a type A derivation from \( G, r \) to \( (\emptyset), r \) in \( P \), where \( G = \leftarrow q_1, \ldots, q_n \). Then \( T(P) \vdash +\partial q_i \) for all \( i = 1, \ldots, n \).

Proof. We use induction on the total length of the type A derivation, that is, on the number of steps in all type A and type B derivations used to establish the top level type A derivation.

Suppose that the top level type A derivation is \( (G_1, r), \ldots, (G_k, r) \), where \( G_1 = G \) and \( G_k = \emptyset \). Consider the first step in this derivation. There must be a rule \( r_1 \) resolving with \( G \) on \( q_1 \). Then \( G_2 \) is the resolvent of \( r_1 \) with \( G \). Suppose that

\[
r_1 = q_1 \leftarrow p_1, \ldots, p_s,
\]

So \( G_2 = \leftarrow p_1, \ldots, p_s, q_2, \ldots, q_n \). Furthermore,

there must be a type B derivation from \( (\leftarrow q_1, r_1) \) to \( (\emptyset, r_1) \). (2)

Note that there is a type A derivation from \( (G_2, r) \) to \( (\emptyset, r) \) which is shorter than the original derivation starting at \( (G_1, r) \). So we can apply the induction hypothesis and get:

\[
T(P) \vdash +\partial p_j \quad \text{for } j = 1, \ldots, s, \quad T(P) \vdash +\partial q_i \quad \text{for } i = 2, \ldots, n.
\]

We complete the proof by showing that we also have \( T(P) \vdash +\partial q_1 \). Conditions (1) and (3) say that there is a rule with head \( q_1 \) such that for all \( a \) in the antecedent of the rule, \( T(P) \vdash +\partial a \). Furthermore \( T(P) \vdash -\Delta \sim q_1 \), since \( T(P) \) includes neither facts nor strict rules. In the following we show that every rule with head \( \sim q_1 \) which is not inferior to \( r_1 \) is not applicable, that is, we can prove \( T(P) \vdash -\partial b \) for at least one antecedent \( b \) of the rule. Then we are finished, by the definition of condition \( +\partial \) in the definition of a derivation in Section 2.\(^1\)

Auxiliary inductive proof: We show that within a successful type A derivation, if there is a B derivation in \( P \) from \( (F, r') \) to \( (\emptyset, r') \), then for every goal \( G \in F \) there is a literal \( q \in G \) with \( T(P) \vdash -\partial q \). Together with (2) above, this completes our proof. In the following we make use of induction on the total length of the type B derivation, that is we count all steps in all type B and type A auxiliary derivations as well. Consider the type B derivation \( (F_1, r'), (F_2, r'), \ldots, (F_n, r') \), where \( F_1 = F \) and \( F_n = \emptyset \).

\(^1\) Ref. [3] showed that the concatenation of derivations is still a derivation. Thus the results \( T(P) \vdash +\partial a \), \( T(P) \vdash -\partial b \) etc. can be combined into one derivation, as is required in the definition of Section 2.
Again we note that we conclude, by clause (2.3) of the hypothesis of the main inductive proof, and conclude that for all such goal top level derivation mentioned in the lemma. Thus we can apply the induction hypothesis (auxiliary induction proof), we conclude that for all \( G \in F_2 \) there is \( q' \) in \( G \) such that \( T(P) \vdash -\partial q' \). By the construction of \( F_2 \) we have

\[
Ar'' \text{ with head } q(r'' \not\in r') \Rightarrow \exists a \in A(r'') : T(P) \vdash -\partial a).
\] (4)

Note that \( T(P) \vdash -\Delta q \), since the strict part of \( T(P) \) is empty. Using this fact and (4) we conclude, by clause (2.3) of the \( -\partial \) condition on proofs (where \( s \) is \( r' \)), that \( T(P) \vdash -\partial q \), which is what we had to prove.

Case 2: Otherwise, let \( \leftarrow q, Q \) be the goal in \( F \) such that \( q \) is selected to be resolved in the next step. According to the definition of a type B derivations, we consider \( R \), the set of rules \( r' \) which resolve with \( \leftarrow q, Q \) on \( q \). Let \( R' \) be the subset of \( R \) containing all rules \( r' \) such that there is not a type A derivation from \( \leftarrow q, r' \) to \( (\emptyset, r') \). Let \( C \) be the set of all resolvents of the rules in \( R' \) with the goal \( \leftarrow q, Q \), resolving on \( q \). Again we note that \( \emptyset \not\in C \) (otherwise we could not derive \( (\emptyset, r') \)); therefore \( F_2 = C \cup (F_1 - \{ \leftarrow q, Q \}) \).

Case 2.1: Suppose that \( R' \neq R \), that is, for at least one rule \( r' \in R \) there is a type A derivation from \( \leftarrow q, r' \) to \( (\emptyset, r') \). This type A derivation is part of the original, top level derivation mentioned in the lemma. Thus we can apply the induction hypothesis of the main inductive proof, and conclude \( T(P) \vdash +\partial \sim q \). By Lemma 4.1 \( T(P) \vdash -\partial q \). Thus the claim of the auxiliary inductive proof is true for the goal \( \leftarrow q, Q \) in \( F_1 \). The other goals in \( F_1 \) are also included in \( F_2 \), by definition. Thus we can apply the induction hypothesis (auxiliary induction proof) and get that in every such goal \( G' \) there is a literal \( q' \) such that \( T(P) \vdash -\partial q' \). Thus the proof is completed in this case.

Case 2.2: \( R = R' \). In this case for every rule \( q \leftarrow B, Q \) is included in \( F_2 \). By the induction hypothesis (auxiliary induction), in every such goal there is at least one literal \( s \) such that \( T(P) \vdash -\partial s \).

Subcase 2.2.1: In at least one of these goals there is a literal \( p \) in \( Q \) such that \( T(P) \vdash -\partial p \). Then we can use this for the goal \( \leftarrow q, A \), too.

Subcase 2.2.2: Otherwise, in each goal \( \leftarrow B, Q \), there must be a literal \( b \) in \( B \) such that \( T(P) \vdash -\partial b \). Therefore there is no rule with head \( q \) such that all its antecedents are provable in \( T(P) \). By the definition of \( -\partial \) in Section 2.2, we conclude \( T(P) \vdash -\partial q \).

In both subcases above we have shown that the claim of the auxiliary induction proof is true for the goal \( \leftarrow q, Q \). Again the other goals in \( F_1 \) are also included in \( F_2 \), by definition. Thus, by the induction hypothesis (auxiliary induction proof), in every such goal \( G' \) there is a literal \( q' \) such that \( T(P) \vdash -\partial q' \). Thus the proof is completed in this case, too. This completes the entire proof. \( \square \)

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.
**Example 3.**

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 \(+\neg\text{mammal}(\text{platypus})\) is indeed provable in defeasible logic: there is a rule in support of *mammal*(*platypus*), and every rule for *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 \(\neg\text{hasBill}(\text{platypus})\)). 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.

We conclude this section with some remarks on the comparison of the relative strength of different nonmonotonic inference systems. Consider the following general argument\(^2\). Suppose that a sceptical nonmonotonic inference system \(L_1\) is strictly stronger than another \(L_2\). Then there is a theory \(T\) and a conclusion \(p\) such that \(p\) can be proven from \(T\) in \(L_1\) but not in \(L_2\). Now add to \(T\) the rules \(q \leftarrow \) and \(\neg q \leftarrow p\). It would appear that in \(L_1\) we cannot prove \(q\) because the second rule fires, while in \(L_2\) we cannot prove \(p\); thus we can prove \(q\). But then \(L_1\) is not stronger than \(L_2\).

This argument is not valid in all cases and, in particular, in discussing the relative strength of defeasible logic and LPwNF as demonstrated by the following example.

**Example 4.** Consider the defeasible theory of Example 3, extended by the following two rules:

\[
\begin{align*}
\text{r}_1 : \text{monotreme}(X) &\Rightarrow \text{mammal}(X) & \text{monotreme}(\text{platypus}) \\
\text{r}_2 : \text{hasFur}(X) &\Rightarrow \text{mammal}(X) & \text{hasFur}(\text{platypus}) \\
\text{r}_3 : \text{laysEggs}(X) &\Rightarrow \neg \text{mammal}(X) & \text{laysEggs}(\text{platypus}) \\
\text{r}_4 : \text{hasBill}(X) &\Rightarrow \neg \text{mammal}(X) & \text{hasBill}(\text{platypus}) \\
\text{r}_1 > \text{r}_3 \\
\text{r}_2 > \text{r}_4
\end{align*}
\]

\(^2\) We thank an anonymous reviewer for suggesting a discussion of this point.
As expected, we cannot derive \( q \) in defeasible logic because \( r_6 \) is applicable and not weaker than \( r_5 \). But neither can we prove \( q \) from the corresponding LPwNF program. The only possible proof of \( q \) would involve rule \( r_5 \) which is attacked by a B-derivation which involves \( r_6 \) and one of \( r_1 \) and \( r_2 \) (with instantiation \( \text{platypus} \)), say \( r_1 \). Since there is no rule stronger than \( r_1 \) the attack succeeds and the proof of \( q \) fails.

A similar situation occurs in the nonmonotonic semantics of logic programs where, for example, the well-founded semantics is stronger than Kunen’s semantics.

5. Other approaches

Courteous logic programs [6] 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 using auxiliary predicates (a technique often used in logic programming, e.g. [7]): Every rule

\[
\begin{align*}
  r : L & \leftarrow L_1 \land \cdots \land L_n \land \neg \text{fail} \ M_1 \land \cdots \land \neg \text{fail} \ M_k
\end{align*}
\]

can be replaced by the rules:

\[
\begin{align*}
  r : L & \leftarrow L_1 \land \cdots \land L_n \land p_r \\
  p_r & \leftarrow \\
  \neg p_r & \leftarrow M_1 \\
  \cdots & \\
  \neg p_r & \leftarrow M_k
\end{align*}
\]

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 \( df(C) \models +\partial q \).

It is worth mentioning that despite the underlying assumption of acyclicity, courteous logic programs are not a special case of LPwNF, because they do incorporate the idea of a team. For example, consider the program in Example 3. \( \text{mammal}(\text{platypus}) \) can be proven in courteous logic programs, but not in LPwNF.

Priority logic [12,13] 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 \( R_1 \) and \( R_2 \) (see [12,13] 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.

**Example 5.**
\[
\begin{align*}
  r_1 & : \text{quaker} \leftarrow \\
  r_2 & : \text{republican} \leftarrow \\
  r_3 & : \text{pacificist} \leftarrow \text{quaker} \\
  r_4 & : \neg \text{pacificist} \leftarrow \text{republican} \\
  r_5 & : \text{footballfan} \leftarrow \text{republican} \\
  r_6 & : \text{antimilitary} \leftarrow \text{pacificist} \\
  r_7 & : \neg \text{antimilitary} \leftarrow \text{footballfan}
\end{align*}
\]

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 \( R_1 \), 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, \( T \Vdash +\partial \text{pacificist} \), so the approaches are different.

And if we use the extensibility relation \( R_2 \), then priority logic does not allow one to prove \( \neg \text{antimilitary} \). But defeasible logic can prove \( +\partial \neg \text{antimilitary} \). 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 [11]).
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” while priority logic takes a fundamentally different approach, which is evident in its propagation of ambiguity.

Of the three formalisms in the same 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.

These results indicate that defeasible logic deserves more attention. In other papers [2,8] we have studied the logic as a formal system, including representation results, properties of the inference relation, and semantics.

References