Relating Defeasible Logic to the Well-Founded Semantics for Normal Logic Programs

Frederick Maier and **Donald Nute**
Department of Computer Science and Artificial Intelligence Center
The University of Georgia
Athens, GA 30602
fmaier@uga.edu and dnute@uga.edu

Abstract

The most recent version of defeasible logic (Nute 1997) is related to the well-founded semantics by translating defeasible theories into normal logic programs. A correspondence is shown between the assertions of a defeasible theory and the literals contained in the well-founded model of the translation. The translation scheme is based upon (Antoniou & Maher 2002) but is modified to account for unique features of this defeasible logic.

Introduction

This paper relates the most recent version of defeasible logic first described in (Nute 1997) to the well-founded semantics for normal logic programs via a translation scheme. After the scheme is presented, a correspondence is proven to exist between, on the one hand, positive and negative assertions of a defeasible theory, and (selected) positive and negative literals of the logic program's well-founded model.

The translation scheme is based upon work of Antoniou and Maher in (2002) which is based on a notational variant of an earlier version of defeasible logic described in (Nute 1992; 1994). For the purposes of this paper, we will call the earlier version of defeasible logic DL and the newer version NDL. Antoniou and Maher show a relationship between DL and stable models, but the result is limited to defeasible theories without cycles in their rules. They establish a more general relationship between all defeasible theories and Kunen's 3 valued semantics described in (Kunen 1987). Antoniou and Maher cite extensive recent work on the theory and applications of DL, and further discussion of applications of NDL can be found in (Nute 2001). A deontic version of NDL is also presented in (Nute 1997).

NDL goes beyond DL by including a more extensive treatment of the ways that defeasible rules may conflict and by explicitly considering failures of proofs due to cycles in defeasible rules. Both of these are important improvements in defeasible logic, and the impact of these changes are discussed in detail in (Nute 2001). Studying NDL in relation to the well-founded semantics is a natural choice. Both are formalisms for nonmonotonic reasoning which explicitly consider dependency cycles between literals, and both are purportedly computationally tractable.

The majority of this paper is devoted to proving the correspondence between the assertions of a defeasible theory in NDL and the well-founded model of the translation. We interpret this result as showing that well-founded models indirectly provide a semantics for NDL. (For an alternative semantics for NDL, see (Donnelly 1999).) Furthermore, our result allows us to use our translation method together with a proof method that is sound with respect the the well-founded semantics as an implementation of NDL.

Defeasible Logic: Language and Proof System

This section presents necessary terminology for NDL and its proof system. For a fuller discussion, see (Nute 2001). The only well-formed formulas in the language for NDL are literals (atomic sentences and their negations). The language also contains *strict* rules, written $A \rightarrow p$ (where p is a literal and A a finite set of literals), *defeasible* rules (written $A \Rightarrow p$), and *undercutting defeaters* or simply *defeaters* (written $A \leadsto p$).

Definition 1: A defeasible theory D is a quadruple $\langle F, R, C, \prec \rangle$, where F is a possibly empty set of literals in the language of D, R a set of rules, C a set of finite sets of literals in the language of D such that for any literal p in the language of the theory, $\{p, \neg p\} \in C$, and $\prec$ an acyclic superiority relation over R.

The basic ideas behind the proof theory for NDL is that we can derive a literal p from a defeasible theory just in case p is one of the initial facts of the theory ($p \in F$,) or p is the head of some strict or *undefeated* defeasible rule in the theory and all of the literals in the body of the rule are also derivable. The role of defeaters is solely to defeat other arguments that might otherwise establish a literal.

Let $D = \langle F, R, C, \prec \rangle$ be a defeasible theory. If C only contains sets of the form $\{p, \neg p\}$, we say D has a minimal conflict set. We say that the conflict set C of D is closed under strict rules if, for all $cs \in C$, if $A \rightarrow p$ is a rule and $p \in cs$, then $\{A \cup (cs - \{p\})\} \in C$. It is not a necessary condition that a defeasible theory be closed under strict rules, but it is certainly an attractive condition. We will call a defeasible theory *closed* if its conflict set is closed under strict rules.

The proof theory for NDL is based upon argument trees:

Definition 2: Let D be a defeasible theory and p a literal in the language of D. The expression $D \mid\!\sim p$ is called a *positive defeasible assertion*, while $D \sim\!\mid p$ is called a *negative defeasible assertion*.

Informally, $D \mathrel{|\!\sim} p$ and $D \mathrel{\sim\!|} p$ are interpreted to mean that a demonstration (respectively, a refutation) exists for p from D. Note that $D \mathrel{\sim\!|} p$ is equivalent to neither $D \mathrel{|\!\sim} \neg p$ nor $D \not\mathrel{|\!\sim} p$. $D \mathrel{\sim\!|} p$ means that there is a demonstration that there is no defeasible proof of p from D.

Definition 3: τ is a *defeasible argument tree* for D iff τ is a finite tree such that every node of τ is labelled either $D \mathrel{|\!\sim} p$ or $D \mathrel{\sim\!|} p$ (for some literal p appearing in D).

Definition 4: The *depth* of a node n is k iff n has $k-1$ ancestors in τ. The depth of a tree is taken to be the greatest depth of any of its nodes.

Definition 5: Let A be a set of literals, and n a node of τ:

1. A *succeeds* at n iff for all $q \in A$, there is a child m of n such that m is labelled $D \mathrel{|\!\sim} q$.
2. A *fails* at n iff there is a $q \in A$ and a child m of n such that m is labelled $D \mathrel{\sim\!|} q$.

Definition 6: τ is a defeasible proof iff τ is an argument tree for D, and for each node n of τ, one of the following obtains:

1. n is labelled $D \mathrel{|\!\sim} p$ and either:
   a. $p \in F$,
   b. there is a strict rule $r : A \rightarrow p \in R$ such that A succeeds at n, or
   c. there is a defeasible rule $r : A \Rightarrow p \in R$ such that A succeeds at n and for all $cs_i \in C$, if $p \in cs_i$, then there is a $q \in (cs_i - (F \cup \{p\}))$ such that
      i. for all strict rules $s : B \rightarrow q \in R$, B fails at n, and
      ii. for all defeasible rules $s : B \Rightarrow q \in R$, either B fails at n or else $s \prec r$, and
      iii. for all defeaters $s : B \leadsto q \in R$, either B fails at n or else $s \prec r$.
2. n is labelled $D \mathrel{\sim\!|} p$ and:
   a. $p \notin F$,
   b. for all strict rules $r : A \rightarrow p \in R$, A fails at n, and
   c. for all defeasible rules $r : A \Rightarrow p \in R$, either
      i. A fails at n, or
      ii. there is a $cs_i \in C$ such that $p \in cs_i$, and for all $q \in cs_i - (F \cup \{p\})$, either
         A. there is a strict rule $s : B \rightarrow q \in R$ such that B succeeds at n,
         B. there is a defeasible rule $s : B \Rightarrow q \in R$ such that B succeeds at n and $s \not\prec r$, or
         C. there is a defeater rule $s : B \leadsto q \in R$ such that B succeeds at n and $s \not\prec r$.
3. n is labelled $D \mathrel{\sim\!|} p$ and n has an ancestor m in τ such that m is labelled $D \mathrel{\sim\!|} p$ and all nodes between n and m are negative defeasible assertions.

Condition 6.3 is called *failure-by-looping*. Since conclusions cannot be established by circular arguments, failure-by-looping can be used to help show that a literal cannot be derived from a defeasible theory.

Definition 7: Where D is a defeasible theory and S is a set of literals in the language of D, $D \mathrel{|\!\sim} S$ if and only if for all $p \in S$, $D \mathrel{|\!\sim} p$.

Some important formal properties of this logic are established in the following theorems.

Theorem 1 (Coherence): *If D is a defeasible theory and $D \mathrel{|\!\sim} p$, then $D \not\mathrel{\sim\!|} p$.*

Theorem 2 (Consistency): *If $D = \langle F, R, C, \prec \rangle$ is a defeasible theory, $S \in C$, and $D \mathrel{|\!\sim} S$, then $\langle F, \{A \rightarrow p : A \rightarrow p \in R\}, C, \prec \rangle \mathrel{|\!\sim} S$.*

Theorem 3 (Cautious Monotony): *If $D = \langle F, R, C, \prec \rangle$ is a defeasible theory, $D \mathrel{|\!\sim} p$, and $D \mathrel{|\!\sim} q$, then $\langle F \cup \{p\}, R, C, \prec \rangle \mathrel{|\!\sim} q$.*

Theorem 1 assures us that we can not both prove and demonstrate the absence of any proof for the same literal. Theorem 2 says that any incompatible set of literals derivable from a defeasible theory must be derivable from the facts and the strict rules alone. In other words, the defeasible rules of a theory can never introduce any new incompatibilites. Of course, this interpretation of Theorem 2 assumes that all possible incompatibilities are captured in the conflict set of the theory. Cautious Monotony is a principle which many authors working on nonmonotonic reasoning propose as a necessary feature for any adequate nonmonotonic formalism.

The advantages of adding failure-by-looping to our proof theory should be obvious. Consider a simple defeasible theory like the following:

$D = \langle \{mammal\}, \{\{furry, has\_wings\} \Rightarrow bat, bat \Rightarrow furry, bat \Rightarrow has\_wings, bat \Rightarrow flies, mammal \Rightarrow \neg flies\}, \{\{bat, \neg bat\}, \{furry, \neg furry\}, \{has\_wings, \neg has\_wings\}, \{mammal, \neg mammal\}, \{flies, \neg flies\}\}, \emptyset \rangle$.

In earlier versions of defeasible logic that lacked failure-by-looping, although we could easily see that there was no way to show $D \mathrel{|\!\sim} bat$, we could not demonstrate this in the proof theory, that is, we could not show $D \mathrel{\sim\!|} bat$. Consequently, neither could we show $D \mathrel{|\!\sim} \neg flies$. Failure-by-looping provides a mechanism for showing $D \mathrel{\sim\!|} bat$, which then allow us to show $D \mathrel{|\!\sim} \neg flies$. When we later define a translation of defeasible theories into a standard logic programs in such a way that the consequences of a theory correspond to the well-founded semantics for the logic program, this example will also serve to show that failure-by-looping is necessary to get this correspondence. Where the theory D above is translated into the standard logic program P_D, `¬defeasibly(bat)`, `¬defeasibly(furry)`, `¬defeasibly(has_wings)`, and `defeasibly(neg_flies)` are all in the well-founded model of P_D, but the corresponding results are undetermined in versions of defeasible logic without failure-by-looping. Where a defeasible theory has cyclic rules, failure-by-looping is needed to capture within the proof theory the concept of a literal being unfounded.

Adding explicit conflict sets and closing them under strict rules provides an alternative solution to a class of examples that have always seemed odd to the authors. Consider the theory

$D = \langle \{quaker, republican\}, \{quaker \Rightarrow dove,$

$republican \Rightarrow hawk$, $dove \Rightarrow activist$, $hawk \Rightarrow activist$, $dove \rightarrow \neg hawk$, $hawk \rightarrow \neg dove\}$, $\{\{quaker, \neg quaker\}$, $\{republican, \neg republican\}$, $\{dove, \neg dove\}$, $\{hawk, \neg hawk\}$, $\{dove, hawk\}\}, \emptyset\rangle$.

Theories like Reiter's default logic (Reiter 1980) would generate two default extensions for a theory like this: $\{quaker, republican, dove, activist\}$ and $\{quaker, republican, hawk, activist\}$. The skeptical approach would accept the intersection of these extensions, $\{quaker, republican, activist\}$, including $activist$ as a *floating conclusion* (Makinson & Schechta 1991). This seems unintuitive to us. A Republican quaker might be a dove or a hawk, but might just as well be neither. This is reflected in NDL. In this theory, the rules $quaker \Rightarrow dove$ and $republican \Rightarrow hawk$ conflict with each other since $\{dove, hawk\}$ is a conflict set in the theory. Neither rule takes precedence over the other; so neither consequent is defeasbily derivable. Thus neither of the rules to establish $activist$ is satisfied, and $activist$ is also not defeasibly derivable. Our proof theory avoids these floating conclusions in an intuitively reasonable way.

Logic Programs

Recall that a logic program consists of a set of rules having the form

$$P :- Q_1, Q_2 \ldots Q_n.$$

where each P and Q_i are atomic formulae of first order logic, or else such formulae preceded by a negation symbol. A rule in which the set of Q's is empty is called a *fact*. The neck ':-' is interpreted as 'if', though it is incorrect to view it as material implication (Gelder, Ross, & Schlipf 1991). Commas separating formulae in the body indicate logical conjunction.

Constants, variables, and function symbols are allowed to appear in each formula. If variables are used, then the rule is assumed to be universally quantified, though the quantifiers themselves are usually omitted in the written program.

Definite logic programs do not contain negation (in any form) in either the head or body of a rule. A *normal* (sometimes *general* (Gelder, Ross, & Schlipf 1991)) logic program may contain atoms in the body preceded by `not`, where the negation is taken to be negation-as-failure: An atom preceded by *not* is true only if it cannot be proven from the program. This can be contrasted with *strong* (sometimes *explicit*) negation (e.g., $\neg p$). Programs in which both strong negation and negation-as-failure appear are called *extended logic programs*. In a normal program, only `not` is allowable.

Definite programs have a unique minimal Herbrand model, and this is often taken to be the intended meaning of the program (Gelder, Ross, & Schlipf 1991). This result was shown in a paper by Van Emden and Kowalski in 1976 (Emden & Kowalski 1976). It was also shown there that this least Herbrand model corresponds to least fixed-point obtainable via the T operator (the *immediate consequence operator*), defined below.

Let P be a logic program and $\mathcal{I}$ a set of ground atoms. Then:

$$T_P(\mathcal{I}) = \{p \mid r \text{ is a rule of } P \text{ with head } p\text{, and each } q_i \text{ in the body of } r \text{ is in } \mathcal{I}\}.$$

Importantly, when negation occurs in a program, a single least Herbrand model need not exist. For instance, the program `p :- not q` has two minimal models: $\{p\}$ and $\{q\}$ (Gelder, Ross, & Schlipf 1991). In such cases, the meaning of the program is unclear.

The Well-Founded Model for Normal Logic Programs

Several attempts have been made to provide a reasonable interpretation of logic programs containing negation. The *well-founded semantics* (Gelder, Ross, & Schlipf 1988; 1991) was developed for normal programs but has since been applied to extended logic programs.

Let P be a normal logic program containing only ground terms. An interpretation $\mathcal{I}$ of P is simply a consistent set of positive and negative literals whose atoms are taken from the Herbrand base of P. The consistency of $\mathcal{I}$ is required—if a literal p is in $\mathcal{I}$, then its complement $\neg p$ does not appear in $\mathcal{I}$.

If p appears in $\mathcal{I}$, then p is said to be $true$ in $\mathcal{I}$. If $\neg p$ appears in $\mathcal{I}$, then p is *false* in $\mathcal{I}$. If neither p nor $\neg p$ appears in $\mathcal{I}$, then p is said to be *undefined* in $\mathcal{I}$.

A set of literals S from the Herbrand base of P is said to be *unfounded* with respect to an interpretation $\mathcal{I}$ iff for each each $p \in S$ and for each rule r of P with head p, there exists a (positive or negative) subgoal q of r such that

1. q is false in $\mathcal{I}$ (that is, $\neg q$ appears in $\mathcal{I}$), or
2. q is positive and appears in S.

Unfounded sets are closed under union, and so for any P and $\mathcal{I}$, there exists a *greatest unfounded set* of P wrt $\mathcal{I}$, denoted $U_P(\mathcal{I})$:

$$U_P(\mathcal{I}) = \{\bigcup A \mid A \text{ is an unfounded set of } P \text{ with respect to } \mathcal{I}\}.$$

$U_P(\mathcal{I})$ can be viewed as a monotone operator and together with T_P is used to define another operator, W_P:

$$W_P(\mathcal{I}) = T_P(\mathcal{I}) \cup \neg \cdot U_P(\mathcal{I})$$

where $\neg \cdot U_P(\mathcal{I})$ is the element-wise negation of $U_P(\mathcal{I})$.

$U_P(\mathcal{I})$, $T_P(\mathcal{I})$, and $W_P(\mathcal{I})$ are all monotonic, and can be used to define the sequence $(\mathcal{I}_0, \mathcal{I}_1, \ldots)$, as follows:

1. $\mathcal{I}_0 = \emptyset$
2. $\mathcal{I}_{k+1} = W_P(\mathcal{I}_k)$

The well-founded model of P, $wfm(P)$, is the limit of this sequence. If the Herbrand base is finite, then this limit can be reached in a finite number of iterations.

In the following, we will only be dealing with the propositional case—only finite grounded defeasible theories and programs are considered. Also, we allow $\mathcal{I}_0$ to contain facts of programs.

Translating a defeasible theory into a logic program

Let D be a finite propositional defeasible theory $\langle F, R, C, \prec \rangle$, where F is a set of facts, R is a set of strict, defeasible, and defeater rules, C is a conflict set, and $\prec$ is an acyclic superiority relation over R.
The translation of D into a normal logic program $LP(D)$ is shown below. We shall use $head(r)$ to denote the head of a rule and $body(r)$ to denote the set of its subgoals. $\neg p$ denotes the complement of a literal p. The expression not indicates negation as failure.

1. For each literal $p \in F$ add the following fact to $LP(D)$:

   ```
   defeasibly(p).
   ```

   Facts cannot be defeated, and so they can always safely be inferred.

2. For every pair of rules r and s, if $r \prec s$, add the following fact to $LP(D)$:

   ```
   sup(s,r).
   ```

3. For each strict rule $r \in R$ ($r : q_1, q_2 \ldots q_n \rightarrow p$), add to $LP(D)$ the following rule:

   defeasibly(p) :- defeasibly(q_1), ... defeasibly(q_n).

   The consequent of a strict rule can be derived if it's antecedent holds. Counterarguments need not be considered.

4. For each defeasible rule $r \in R$ ($r : q_1, q_2 \ldots q_n \Rightarrow p$), add to $LP(D)$ rules of the following form:

   a. defeasibly(p) :- defeasibly(q_1), ... defeasibly(q_n), ok(r).
   A literal p is defeasibly provable using r if each subgoal $q_i \in body(r)$ is defeasibly provable and it's *ok* to detach the head (i.e., the rule is not blocked). For each defeasible rule in D, exactly one rule of this form is added to $LP(D)$.

   b. ok(r) :- ok(r, cs$_1$), ok(r, cs$_2$)... ok(r, cs$_m$).
   where $head(r) \in cs_i$. It's ok to apply rule r if it's ok with respect to each conflict set containing the head of r (note that there will always be at least one conflict set).

   c. ok(r, cs$_i$) :- blocked_literal(r, q$_j$).
   where $q_j \in cs_i - (F \cup head(r))$. It's ok to apply rule r with respect to a conflict set if there's a blocked literal in the set $cs_i - (F \cup head(r))$. Note that a rule of this form will exist for each $q_j \in cs_i - (F \cup head(r))$. Facts need not be considered, since these can never be blocked.

   d. blocked_literal(r, q$_i$) :-
      blocked(r,r_1),
      blocked(r,r_2),
      ...
      blocked(r,r_k).
   where $head(r_1) = head(r_2) \ldots = head(r_k) = q_i$, and each r_i is any sort of rule. A literal q_i is blocked with respect to a rule r if every rule r_i having it as a head is blocked by r.
   One clause of the above form will exist for each unique (r, q_i) pair, where $q_i \in cs_i - (F \cup head(r))$ and cs_i is a conflict set containing $head(r)$. The literal q_i can never be a fact of the defeasible theory, since (again) facts cannot be blocked. Also, if no rule has head q_i, then blocked_literal(r, q$_i$) is asserted as a fact of the logic program.

   e. blocked(r,r$_i$) :- not defeasibly(s$_j$).
   where $s_j \in body(r_i)$. A rule r blocks another r_i if r_i has a subgoal s_j that is not defeasibly provable.
   If a rule r_i has no body (in the case of defeasible rules), then no clauses of form (e) will occur in $LP(D)$.

   f. blocked(r,r$_i$) :- sup(r,r$_i$).
   A rule r also blocks another r_i if r is takes precedence over r_i.

Note that the only place negation-as-failure occurs in the translation is in 3.e. Also, for any strict or defeasible rule r of D with head p, there is a single corresponding logic program rule with head defeasible(p). We refer to this corresponding rule as $trans(r)$.
In the following section, we show that the above translation process is correct, in that $D \mid\!\sim p$ iff defeasible(p) $\in wfm(LP)$, and $D \mid\!\sim p$ iff $\neg$defeasible(p) $\in wfm(LP)$.

A Proof of the Translation's Correctness

We begin with the 'if' direction and induct on the depth of a defeasible argument tree τ.
Let D be a defeasible theory, LP its logic program translation, and τ be a defeasible argument tree for D.

Theorem 4: *If the root of τ is labeled with $D \mid\!\sim p$, then* defeasibly(p) $\in wfm(LP)$. *If the root of τ is labeled with $D \sim\!\mid p$, then* $\neg$defeasibly(p) $\in wfm(LP)$.

(Base Case) Suppose τ is just a single node n labeled $D \mid\!\sim p$ or $D \sim\!\mid p$. We consider each case separately.

Case 1: Suppose that n is labeled $D \mid\!\sim p$. Then definition 6.1 holds.
Since strict rules with empty bodies are forbidden, then either 6.1.a or 6.1.c must obtain. If 6.1.a obtains, p is a fact of D and defeasibly(p) is a fact of LP, and so obviously defeasibly(p) $\in wfm(LP)$. If 6.1.c obtains, then there is some rule $r : A \Rightarrow p$ that succeeds at n, and for all conflict sets cs such that $p \in cs$, there is a literal $q \in (cs - (F \cup \{p\}))$ such that 6.1.c.i, 6.1.c.ii, and 6.1.c.iii hold. Since n has no children, A must be empty.
Let cs_i be a conflict set such that $p \in cs_i$, and $q_j \in (cs_i - (F \cup \{p\}))$ a literal with the above properties. Since n has no children, there can be no strict rules with q_j as a head. For the same reason, the body of any defeasible or defeater rule with head q_j must be empty.
Let s_k be a defeasible or defeater rule with head q_j. Since 6.1.c obtains, $s_k \prec r$. Generalizing on s_k, for all rules s with head q_j, $s \prec r$, and so sup(r, s) appears as a fact in LP. Given this, blocked_literal(r, q_j) and hence ok(r, cs_i)$\in wfm(LP)$.

Generalizing on cs_i, for each conflict set cs containing p, `ok(r, cs)`$\in wfm(LP)$. But if this is the case, `ok(r)` $\in wfm(LP)$, and so `defeasibly(p)`$\in wfm(LP)$.

Case 2: Suppose that n is labelled $D \sim\!\!| \ p$.
τ consists of only a single node, and so failure-by-looping cannot apply. Definition 6.2 must obtain: p is not a fact, there are no strict rules with head p (since n is a leaf node), and for all defeasible rules $r : A \Rightarrow p$, either

i A fails at n, or

ii There is a conflict set cs_i such that $p \in cs_i$, and for all $q \in (cs_i - (F \cup \{p\}))$, either,

 a There is a strict rule s: $B \rightarrow q$ such that B succeeds at n, or

 b There is a defeasible rule s: $B \Rightarrow q$ such that B succeeds at n and $s \not\prec r$, or

 c There is a defeater s: $B \leadsto q$ such that B succeeds at n and $s \not\prec r$.

Let r_i be a rule $A \Rightarrow p$. Since n is a leaf node, neither i nor $ii.a$ can obtain.
Let cs_k be a conflict set satisfying (ii) above, and let q be a member of $(cs_k - (F \cup \{p\}))$. Suppose (ii.b) or (ii.c) obtains. Then there is a rule s: $B \Rightarrow q$ ($B \leadsto q$) such that B succeeds at n and $s \not\prec r_i$. Since B must be empty, clauses of the form

```
   blocked(r_i, s):- not defeasibly(q)
```

do not appear in LP. Since $s \not\prec r_i$, the clause `sup(r_i,s)` does not appear as a fact in LP, and because no rules can derive `sup(r_i,s)`, $\neg$`sup(r_i,s)`$\in wfm(LP)$. The only rule with head `blocked(r_i, s)` now has a subgoal whose complement appears in $wfm(LP)$, and so $\neg$`blocked(r_i, s)`$\in wfm(LP)$.
Similarly, since `blocked(r_i, s)` appears in the rule

```
blocked_literal(r_i, q):-
     blocked(r_i, s_1),
     blocked(r_i, s_2),
     ...
     blocked(r_i, s_k).
```

and there are no other rules with that head, $\neg$`blocked_literal(r_i, q)`$\in wfm(LP)$.
Generalizing on q, for any q_j in the set $cs_k - (F \cup \{p\})$), $\neg$`blocked_literal(r_i, q_j)`$\in wfm(LP)$. From this it follows that $\neg$`ok(r_i, cs_k)`$\in wfm(LP)$, and hence $\neg$`ok(r_i)`$\in wfm(LP)$.
Generalizing on rule r_i, $\neg$`defeasibly(p)`$\in wfm(LP)$.

(Induction) Suppose the claim holds for trees of depth k or less and let τ be a tree of depth $k + 1$.

Case 1: Suppose the root n of τ is labeled $D \mid\!\sim p$. Then 6.1 again holds. We will show that regardless of whether 6.1.a, 6.1.b, or 6.1.c is true, `defeasibly(p)`$\in wfm(LP)$:

Case 1.a: p is a fact of D, in which case `defeasibly(p)`$\in wfm(LP)$.

Case 1.b: There is a strict rule $A \rightarrow p$ such that A succeeds at n. Then for all $q \in A$, there is a child of m labeled $D \mid\!\sim q$. Each such q is the root of a valid argument tree of maximum depth less than $k + 1$, and so by inductive hypothesis, `defeasibly(q)`$\in wfm(LP)$. Applying the immediate consequence operator, `defeasibly(p)`$\in wfm(LP)$.

Case 1.c: There is a rule r: $A \Rightarrow p$ such that A succeeds at n and for all conflict sets cs with $p \in cs$, there is a q in $cs - (F \cup \{p\})$) such that:

i. For all strict rules s: $B \rightarrow q$, B fails at n,

ii. For all defeasible rules s: $B \Rightarrow q$ fails at n or else $s \prec r$, and

iii. For all defeater rules s: $B \leadsto q$ fail at n or else $s \prec r$.

Let cs_i and q_k be such a conflict set and literal, and let s be any rule with head q_k. If s fails at n, then some some child of n is labeled $D \sim\!\!| \ d$, where d is in the body of s. This is the root of a valid proof tree, and so by inductive hypothesis, $\neg$`defeasibly(d)` $\in wfm(LP)$. Based on this, `blocked(r, s)`$\in wfm(LP)$.
If s is a defeasible or defeater rule and $s \prec r$, then `sup(r,s)` is a fact of LP, and so `blocked(r, s)` is again in $wfm(LP)$.
Generalizing on s, `blocked_literal(r, q`$_k$`)`$\in wfm(LP)$, and so `ok(r, cs_i)` $\in wfm(LP)$. Generalizing on cs_i, `ok(r, cs)` $\in wfm(LP)$ for all cs in which p appears, and so `ok(r)` appears in $wfm(LP)$.
Since A succeeds at n, for all $u \in A$, there is a child of n labeled $D \mid\!\sim u$. Each such u is the root of a valid argument tree of maximum depth less than $k + 1$, and so by inductive hypothesis, `defeasibly(u)` $\in wfm(LP)$.
It is now that case that every subgoal of the LP rule

```
   defeasibly(p) :-
            defeasibly(u_1),
            ...
            defeasibly(u_n),
            ok(r).
```

corresponding to $r \in R_D$ is true in the $wfm(LP)$. It follows that `defeasibly(p)` $\in wfm(LP)$.

Case 2: Suppose the root n of τ is labeled $D \sim\!\!| \ p$.
For this part of the proof, we show that the collection of literals appearing in negative assertions of τ correspond to an unfounded set with respect to $wfm(LP)$.
Any branch of a proof tree involving failure-by-looping need not extend beyond the topmost node where definition 6.3 applies. As this is so, the tree can be trimmed to that point, and so 6.3 only applies to the leaves of the tree. We may assume without loss of generality that τ is of this form.
Define S and U as follows:

$S = \{n : n$ is a node of τ labeled with $D \sim\!\!| \ u$ for some $u\}$.

$U = \{$`defeasibly(u)`: $D \sim\!\!| \ u$ appears as a label for some member of $S\}$.

Let n be any node in S. Then n is labeled $D \sim\!\!| \ \phi$ for some ϕ. The following shows that all rules with head ϕ are unfounded.
Node n is either a leaf or an internal node.

Case 2.a: Suppose that n is an internal node. Then 6.2 obtains, and ϕ is not a fact.

If r is a strict or defeasible rule with head ϕ that fails at n, n has a child labeled $D \sim\!\!| \; v$, where $v \in body(r)$. By definition `defeasibly(v)` appears in U.
If r is a defeasible rule with head ϕ that does not fail at n, then by 6.2.c.ii there is a $cs \in C$ such that $p \in cs$, and for all $q \in cs - (F \cup \{p\})$, either

i. there is a strict rule $s : B \rightarrow q \in R$ such that B succeeds at n,

ii. there is a defeasible rule $s : B \Rightarrow q \in R$ such that B succeeds at n and $s \not\prec r$, or

ii. there is a defeater rule $s : B \rightsquigarrow q \in R$ such that B succeeds at n and $s \not\prec r$

Let cs_i be such a conflict set as above, q_j a member of $cs_i - (F \cup \{p\})$, and let s be a strict (defeasible, defeater) rule with head q_j such that $body(s)$ succeeds at n and $s \not\prec r$ (even if s is strict, $s \not\prec r$ holds).
Since $body(s)$ succeeds at n, n has a child labeled $D \mid\!\sim u$ for each $u \in body(s)$. By inductive hypothesis, `defeasibly(u)` $\in wfm(LP)$ for each $u \in body(s)$. Since $s \not\prec r$, ¬`sup(r,s)` $\in wfm(LP)$. As this is so, every rule with head `blocked_(r,s)` has a subgoal whose complement appears in $wfm(LP)$, and so ¬`blocked_(r,s)` itself appears in $wfm(LP)$.
Since ¬`blocked_(r,s)` $\in wfm(LP)$, ¬`blocked_literal(r,q)` $\in wfm(LP)$. Generalizing on q, ¬`ok(r,cs_i)` $\in wfm(LP)$, and hence ¬`ok(r)` $\in wfm(LP)$.
Generalizing on rule r, every logic program rule with head `defeasibly(`ϕ`)` has a subgoal in U or else a subgoal whose complement appears in $wfm(LP)$.

Case 2.a: Suppose that n is a leaf node. Either 6.2 or 6.3 obtains. If 6.2 obtains, then as was shown in the base case, for any rule r with head `defeasibly(`ϕ`)`, ¬`ok(r)` $\in wfm(LP)$. If 6.3 obtains, then there is a non-leaf node labeled $D \sim\!\!| \; \phi$, and we have shown there that all rules with head `defeasibly(`ϕ`)` have (1) a subgoal whose complement appears in $wfm(LP)$, or (2) a subgoal in U.

Given the above 2 cases, for every member `defeasibly(u)` $\in U$, each rule with head `defeasibly(u)` has a subgoal in U or else a subgoal whose complement is in $wfm(LP)$. By definition, U is unfounded with respect to the $wfm(LP)$.
As this is so, for each $u \in U$, ¬`defeasibly(u)` $\in wfm(LP)$. In particular, ¬`defeasibly(p)` $\in wfm(LP)$. □

For the 'only if' portion of the proof, it is convenient to transform the logic program by combining rules of the form

```
ok(r) :-
    ok(r, cs_1),
    ok(r, cs_2),
    ...
    ok(r, cs_k).
ok(r, cs_i) :-
    blocked_literal(r, q_1).
```

to create new rules such as below:

```
ok(r) :-
    blocked_literal(r, q_1),
    ...
    blocked_literal(r, q_k).
```

The old rules are deleted from the program. In each new rule, the i^{th} `blocked_literal(r, q)` corresponds to one element of the set $cs_i - (F \cup \{head(r)\})$. Intuitively, this means that rule r is 'ok' to fire if each conflict set has a blocked literal. Each rule contains a tuple from $\{cs_1 - (F \cup \{head(r)\})\} \times \{cs_2 - (F \cup \{head(r)\})\} \times \ldots \{cs_k - (F \cup \{head(r)\})\}$. Every possible combination is used in some rule.

The following lemmas are used in the proof:

Lemma 1: *If ok$(r) \in U_n$, then there exists a conflict set cs in D with head$(r) \in cs$, and for each $q \in cs - (F \cup \{head(r)\})$ there exists a rule s of D with head(q) such that $s \not\prec r$ and for each $v \in body(s)$,* `defeasibly(v)` $\in \mathcal{I}_{n-1}$.

Suppose $ok(r) \in U_n$. Then every rule of the form

```
ok(r) :-
    blocked_literal(r, q_1),
    ...
    blocked_literal(r, q_k).
```

has a subgoal ¬`blocked_literal(r, q`$_i$`)` in $\mathcal{I}_{n-1}$ or else in U_n. If ¬`blocked_literal(r, q`$_i$`)`$\in I_{n-1}$, then `blocked_literal(r, q`$_i$`)`$\in U_{n-1}$. U is monotonic, and so we may assume wlog that `blocked_literal(r, q`$_i$`)` $\in U_n$.
Since a tuple from $\{cs_1 - (F \cup \{head(r)\})\} \times \{cs_2 - (F \cup \{head(r)\})\} \times \ldots \{cs_m - (F \cup \{head(r)\})\}$ comprises the body of each rule with head $ok(r)$, and every rule has a subgoal in U_n, it readily follows that for at least one set $cs_i - (F \cup \{head(r)\})$, every literal of $cs_i - (F \cup \{head(r)\})$ is in U_n.
Let q_j be some literal in $cs_i - (F \cup \{head(r)\})$. Since `blocked_literal(r, q`$_j$`)`$\in U_n$, it follows that the rule

```
blocked_literal(r, q_j):-
    blocked(r, s_1),
    blocked(r, s_2),
    . . .
    blocked(r, s_m).
```

has some subgoal `blocked(r, s_u)` $\in U_n$. Rules with head `blocked(r, s_u)` have `not defeasibly(v)` or `sup(r,s_u)` as their bodies, where v is some subgoal of s_u.
Applying the definition of unfoundedness, for each rule with body `not defeasibly(v)`, `defeasibly(v)` must appear in $\mathcal{I}_{n-1}$, and so `defeasibly(v)` $\in \mathcal{I}_{n-1}$ for every $v \in body(s_u)$. If `sup(r,s_u)` appears as the body of some rule, then $sup(r, s_u) \in U_n$ (the same holds even if no rule has such a body). If that is the case, then $s_u \prec r$ does not appear in D.
Generalizing on q_j, every literal of $cs_i - (F \cup \{head(r)\})$ has a rule s with $s \not\prec r$ such that for all $v \in body(s)$, `defeasibly(v)` $\in \mathcal{I}_{n-1}$. □

Lemma 2: *If $ok(r) \in \mathcal{I}_n$, then for each conflict set cs containing $head(r)$, there exists a $q \in cs - (F \cup \{head(r)\})$ such that for all rules s with head q, either: (1) s contains a subgoal* `defeasible(v)` $\in U_{n-1}$, *or (2) $s \prec r$ is in D.*

If `ok(r)` $\in \mathcal{I}_n$, there is some rule

```
ok(r) :-
    blocked_literal(r, u_1),
    ...
    blocked_literal(r, u_k).
```

such that each subgoal is in $\mathcal{I}_m$, for some $m < n$. Each `blocked_literal(r,u_j)` appears as the head of exactly one rule of $LP(D)$:

```
blocked_literal(r, u_j):-
    blocked(r, s_1),
    blocked(r, s_2),
    ...
    blocked(r, s_n).
```

Since `blocked_literal(r,`u_j`)` $\in \mathcal{I}_m$, each subgoal `blocked(r, s_i)` must be in $\mathcal{I}_l$, for some $l < m$.
Rules with head `blocked(r, s_i)` can have `not defeasibly(v)` or `sup(r,s_i)` as their bodies. Since `blocked(r, s_i)` $\in \mathcal{I}_l$, either $\neg$`defeasibly(v)` $\in \mathcal{I}_k$, $k < l$ for some $v \in s_i$ or else `sup(r, s_i)` is a fact of $LP(D)$. If the former, `defeasibly(v)` $\in \mathcal{U}_k$ (and hence in U_{n-1}). If the latter, then $s_i \prec r$ must appear in D.
Generalizing on s_i, for each rule s of D with head u_j, $s \prec r$ or else for some $v \in s$, `defeasibly(v)` $\in U_{n-1}$. Generalizing on cs_i, for each conflict cs set containing $head(r)$, there exists a $u \in cs - (F \cup \{head(r)\})$ such that if rule r_i has head u, then $r_i \prec r$ or else r_i contains a subgoal in U_{n-1}. □

Theorem 5: Let $\mathcal{I}_0 = \{$`defeasible(p)`$\,|\, p \in F_D\}$ and $\mathcal{I}_{k+1} = W_{LP}(\mathcal{I}_k)$. For all $\mathcal{I}_n$, if `defeasibly(p)` $\in \mathcal{I}_n$, then $D \mid\!\sim p$, and if $\neg$`defeasibly(p)` $\in \mathcal{I}_n$, then $D \sim\!\!\mid p$.

(Base Case) Suppose `defeasibly(p)` $\in \mathcal{I}_0$. Then `defeasibly(p)` is a fact of LP and so p is a fact of D. Trivially, $D \mid\!\sim p$.
For any p, $\neg$`defeasibly(p)` $\notin \mathcal{I}_0$, and so the claim is trivially satisfied.

(Induction) Suppose that `defeasibly(p)` $\in \mathcal{I}_k$ implies $D \mid\!\sim p$ and $\neg$`defeasibly(p)` $\in \mathcal{I}_k$ implies $D \sim\!\!\mid p$, for all $k \leq n$. We will treat positive and negative literals in turn.

Case 1: Let `defeasibly(p)` $\in \mathcal{I}_{n+1}$. Then either `defeasibly(p)` is a fact of $LP(D)$ (and so obviously $D \mid\!\sim p$) or else there is some rule $t = trans(r)$ with head `defeasibly(p)` such that for all $q \in body(t)$, $q \in \mathcal{I}_n$.
If each subgoal of t is of form `defeasibly(q_i)`, then r is strict. By inductive hypothesis, $D \mid\!\sim q_i$ for each such q_i, and so for each there exists a defeasible proof tree with root labeled $D \mid\!\sim q_i$ (denoted $\tau_{D \mid\sim q_i}$). We may append these proofs to a node labeled $D \mid\!\sim p$ to form a proof of $D \mid\!\sim p$.
If t contains an additional subgoal $ok(r)$, then r is defeasible. Since `ok(r)` $\in \mathcal{I}_n$, by Lemma 2 it follows that for each conflict cs set containing p, there exists a $u \in cs - (F \cup \{p\})$ such that if rule r_i has head u, then $r_i \prec r$ or else r_i contains a subgoal `defeasible(v)` in U_{n-1}. By inductive hypothesis, a refutation proof exists for each v. Where applicable, we may append these to a root node to show $D \mid\!\sim p$.
Since rule t must correspond to either a defeasible or strict rule of r, we may conclude $D \mid\!\sim p$.

Case 2: Suppose that $\neg$`defeasibly(p)` $\in \mathcal{I}_{n+1}$. Then by definition `defeasibly(p)` $\in U_{n+1}$. It immediately follows that p is not a fact of D.
We construct a series of trees as follows. Let τ_0 be the tree consisting of a single unmarked node labeled $D \sim\!\!\mid p$. For any $n > 0$, pick an unmarked leaf node x in τ_n. From the definition of τ_0 and the cases below, we will see that x is labeled $D \sim\!\!\mid q$ and `defeasibly(q)` $\in U_{n+1}$. Then for each rule r with head q, $trans(r)$ has a subgoal s such that $\neg s \in \mathcal{I}_n$, or else $s \in U_{n+1}$. Either $s =$ `defeasibly(t)` for some $t \in body(r)$ or $s = ok(r)$. Consider each posibility in turn.
Case 2.a: $s = ok(r)$ and $ok(r) \in U_{n+1}$. By Lemma 1, there exists a conflict set cs in D with $q \in cs$, and for each $u \in cs-(F \cup \{q\})$, there exists a rule r' with head u such that $r' \not\prec r$ and `defeasibly(v)` $\in I_n$ for each v in the body of r'. By inductive hypothesis, for each such v, $D \mid\!\sim v$. For each $u \in cs - (F \cup \{q\})$, if its associated rule s has a nonempty body, append proof trees for its body to x and mark every node of each such subtree.
Case 2.b: $s = ok(r)$ and $\neg ok(r) \in \mathcal{I}_n$. By definition, $ok(r) \in \mathcal{U}_n$. Since U is monotonic, $ok(r) \in U_{n+1}$, and we make the same additions to τ_n.
Case 2.c: $s =$ `defeasible(t)` and $\neg$`defeasible(x)` $\in \mathcal{I}_n$. Then `defeasible(t)` $\in U_n$, and by inductive hypothesis $D \sim\!\!\mid t$. Append to x a proof tree for $D \sim\!\!\mid t$ and mark every node of this subtree.
Case 2.d: $s =$ `defeasible(t)` and `defeasible(t)` $\in U_{n+1}$. Append to x a node y labled $D \sim\!\!\mid t$. If y satisfies condition i or 3 in Definition 6, then mark y. Otherwise, leave y unmarked.
After applying one of the cases 2.a-2.d for each rule r with head q, examine the resulting tree to see if there is an unmarked non-leaf node z in the tree such that all the children of z are marked. If such a node z is found, mark it. Repeat this procedure until there are no more unmarked nodes in the tree all of whose children are marked. The resulting tree is τ_{n+1}.
Let $\tau = \bigcup_{i=0}^{\infty} \tau_i$.
Suppose x is a marked node in τ. If x was added to τ using cases 2.a, 2.b, or 2.c, then x occurs within a subtree of τ that is a proof tree. So x must satisfy one of the conditions in Definition 6. If x was added to τ and marked according to case 2.d, then x is a leaf node in τ and x satisfies condition 2 or 3 of Definition 6. Otherwise, x is a non-leaf node in τ, x was added to τ using condition 2.d, and x was marked because all of its children were marked. Looking at the four cases used to add the children of x to τ, we see that x must satisfy condition 2 in Definition 6. So if τ is finite and if every node in τ is marked, then τ is a proof tree.
Since D contains only finitely many rules, the branching fac-

tor for constructing τ must be finite. So if τ is infinite, then τ must have an infinitely long branch. Consider such a branch. Every node in this branch (other than the top node) must have been added using case 2.d since all the other branches add proof trees which are finite. So every node in the branch must be labeled $D \sim\!\mid q$ for some literal q. Furthermore, no node in the branch satisfies condition 3 in Definition 6 since if it did, it would have been marked when it was added to τ and it would therefore have no children. But since D is finite, only finitely many literals occur in D. So there must be some literal q such that two different nodes in our infinite branch are labeled $D \sim\!\mid q$. But then one of these two nodes does satisfy condition 3 of Definition 6, which is a contradiction. Therefore, τ is not infinite.
Since τ is not infinite, we can let n be a non-negative integer such that $\tau = \tau_n$. Suppose τ_n has an unmarked node. Since a node must be marked if all its children are marked, τ_n must have an unmarked leaf node x. This node must have been added by case 2.d of our construction, and so we can let q be a literal such that x is labeled $D \sim\!\mid q$, and `defeasible(q)` $\in U_{n+1}$. Since x is not marked, it satisfies neither condition 2 or 3 of Definition 6. Since `defeasible(q)` $\in U_{n+1}$, $q \notin F$. If there is no rule $r \in R$ such that $head(r) = q$, then x satisfies condition 2 of Definition 6. So there is a rule $r \in R$ such that $head(r) = q$, and one of the cases 2.a-2.d applies to x. So there must be some $m > n$ such x has a child node in τ_m. Then x is not a leaf node in τ_m and x is not a leaf node in τ, a contradiction. Therefore, every node in τ satisfies some condition in Definition 6 and τ is a proof tree.□

References

Antoniou, G., and Maher, M. J. 2002. Embedding defeasible logic into logic programs. In *Proceedings of ICLP*, 393–404.

Donnelly, S. 1999. *Semantics, Soundness, and Incompleteness for a Defeasible Logic*. Masters thesis, The University of Georgia.

Emden, M. H. V., and Kowalski, R. 1976. The semantics of predicate logic as a programming language. *Journal of the ACM* 23:733–742.

Gelder, A. V.; Ross, K. A.; and Schlipf, J. 1988. Unfounded sets and well-founded semantics for general logic programs. In *Proceedings 7th ACM Symposium on Principles of Database Systems*, 221–230.

Gelder, A. V.; Ross, K. A.; and Schlipf, J. 1991. The well-founded semantics for general logic programs. *Journal of the ACM* 221–23.

Kunen, K. 1987. Negation in logic programming. *Journal of Logic Programming* 4:289–308.

Makinson, D., and Schechta, K. 1991. Floating conclusions and zombie paths: two deep difficulties in the 'directly skeptical' approach to inheritance nets. *Artificial Intelligence* 48:199–209.

Nute, D. 1992. Basic defeasible logic. In del Cerro, L. F., and Penttonen, M., eds., *Intensional Logics for Programming*. Oxford University Press. 125–154.

Nute, D. 1994. Defeasible logic. In Gabbay, D., and Hogger, C., eds., *Handbook of Logic for Artificial Intelligence and Logic Programming, Vol. III*. Oxford University Press. 353–395.

Nute, D. 1997. Apparent obligation. In Nute, D., ed., *Defeasible Deontic Logic*, Synthese Library. Dordrecht, Netherlands: Kluwer Academic Publishers. 287–315.

Nute, D. 2001. Defeasible logic: Theory, implementation, and applications. In *Proceedings of INAP 2001, 14th International Conference on Applications of Prolog*, 87–114. Tokyo: IF Computer Japan.

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