Ambiguity Propagating Defeasible Logic and the Well-Founded Semantics

Frederick Maier and Donald Nute

Department of Computer Science and Artificial Intelligence Center
The University of Georgia
Athens, GA 30602
fmaier@uga.edu 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 using a simple scheme proposed in (Brewka, 2001). It is found that by introducing ambiguity propagation into this logic, the assertions of defeasible theories coincide with the well-founded models of their logic program translations. Without this addition, the two formalisms are found to disagree in important cases.
A translation in the other direction is also provided. By treating default negated atoms as presumptions in defeasible logic, normal logic programs can be converted into equivalent defeasible theories.

1 Introduction

This paper relates the most recent version of defeasible logic described in (Nute, 1997) to the well-founded semantics for normal logic programs via a translation scheme. After the scheme is presented, it is shown that in important cases the conclusions of a given defeasible theory do not correspond with the logic program's well-founded model. However, by modifying the proof system of defeasible logic, a new ambiguity propagating variant is created, and for this new variant the correspondence holds.

The translation scheme used here was first proposed in (Brewka, 2001) and was used to compare logic programs under a prioritized well-founded semantics to a variant of defeasible logic presented in (Antoniou *et al.*, 2000). For the purposes of this paper, we will call the defeasible logic presented in (Nute, 1997) NDL to distinguish it from other variants. The ambiguity propagating variant presented here will be called ADL. NDL goes beyond the variants presented in (Antoniou *et al.*, 2000) 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 rules. Both of these are important improvements in defeasible logic, and the impact of these changes are discussed in detail in (Nute, 2001).

Another translation scheme for NDL is found in (Maier & Nute, 2006). Defeasible theories are translated into logic programs via the introduction of new literals explicitly representing rules and conflicts. The scheme has the virtue of allowing one to successfully embed defeasible theories of NDL into logic programs, but it does not reveal any deep connection between defeasible logic and logic programming. The translation

there is in turn based upon one found in (Antoniou & Maher, 2002) which uses a notational variant of an earlier version of defeasible logic described in (Nute 1992, 1994). Antoniou and Maher show a relationship between this variant 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 three-valued semantics (Kunen, 1987).

The main contribution of this paper is the development of the ambiguity propagating defeasible logic ADL and a demonstration of a correspondence between the assertions of ADL theories and the well-founded models of the translations. The well-founded semantics can thus be viewed as indirectly providing a semantics for ADL. This result allows us to use the translation method together with a proof method that is sound with respect to the well-founded semantics as an implementation of ADL.

Furthermore, it is shown here that normal logic programs can be translated into ADL theories in such a way that the well-founded model of a given logic program corresponds to the assertions of its translation. Each default negated literal '$not\ p$' of the program is treated as a defeasible rule $\emptyset \Rightarrow \neg p$ of the defeasible theory (stating, in effect, that $\neg p$ presumably holds).

2 The Defeasible Logics NDL and ADL

This section presents the language of NDL and its ambiguity propagating counterpart ADL as well as proof systems for each. NDL is presented first, and ADL is presented as a modification of NDL's proof system. For a fuller discussion of NDL, see (Nute, 2001).

The well-formed formulas in the common language of NDL and ADL 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 $A \rightsquigarrow p$). In the following we will use $\dashrightarrow$ to stand for any rule, whether strict, defeasible, or defeater. Where A is the empty set or a singleton, we will often omit the braces (writing $\Rightarrow p$ instead of $\{\} \Rightarrow p$).

The basic idea behind the proof theory for NDL is that we can derive a literal p from a defeasible theory just in case 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. A strict rule with an empty body is called a *fact* and the head of such a rule is by definition always derivable. A defeasible rule with an empty body is called a *presumption* and may be defeated by other rules. The role of defeaters is solely to defeat other arguments that might otherwise establish a literal. E.g., the defeater $q \rightsquigarrow \neg p$ can be used to prevent proving p, but it cannot be used to directly prove $\neg p$.

Definition 1: A *defeasible theory* D is a triple $\langle R, C, \prec \rangle$, where R is a finite set of rules, each with a possibly empty antecedent, C a set of finite sets of literals in the language of D such that for any literal p appearing in D, $\{p, \neg p\} \in C$, and $\prec$ an acyclic binary superiority relation over the non-strict rules in R.

Let $D = \langle R, C, \prec \rangle$ be a defeasible theory. The sets of literals in C are called *conflict sets*. Each conflict set $cs \in C$ specifies a set of literals that cannot simultaneously consistently hold. 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 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 appearing in D. The expression $D \mid\sim p$ is called a *positive defeasible assertion*, while $D \sim\mid p$ is called a *negative defeasible assertion*. When it is clear which theory is in question, the assertions may be abbreviated $\mid\sim p$ and $\sim\mid p$

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

Definition 3: τ is an *argument tree* for D iff τ is a finite tree such that every node of τ is labeled either $\mid\sim p$ or $\sim\mid p$ (for some literal p appearing in D).

Definition 4: The *depth* of a node n is k iff n has k 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 a defeasible argument tree τ:

1. A *succeeds* at n iff for all $q \in A$, there is a child m of n such that m is labeled $\mid\sim q$.
2. A *fails* at n iff there is a $q \in A$ and a child m of n such that m is labeled $\sim\mid q$.

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

1. n is labeled $\mid\sim p$ and either:
   a. there is a strict rule $r : A \rightarrow p \in R$ such that A succeeds at n, or
   b. there is a defeasible rule $r : A \Rightarrow p \in R$ such that A succeeds at n and for all $cs \in C$, if $p \in cs$, then there is a $q \in cs - \{p\}$ such that for all rules $s : B \dashrightarrow q \in R$, either B fails at n or else $s \prec r$.
2. n is labeled $\sim\mid p$ and:
   a. for all strict rules $r : A \rightarrow p \in R$, A fails at n, and
   b. for all defeasible rules $r : A \Rightarrow p \in R$, either
      i. A fails at n, or
      ii. there is a $cs \in C$ such that $p \in cs$, and for all $q \in cs - \{p\}$, there is a rule $s : B \dashrightarrow q \in R$ such that B succeeds at n and $s \not\prec r$.
3. n is labeled $\sim\mid p$ and has an ancestor m in τ labeled $\sim\mid p$, 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 help to show that a literal cannot be derived from a defeasible theory.

Definition 7: Where D is a defeasible theory and p is a literal appearing in D, $D \mid\sim_{NDL} p$ iff there is a defeasible proof τ in NDL such that the root of τ is labeled $\mid\sim p$.

$D \sim|_{NDL} p$ iff there is a defeasible proof τ such that the root of τ is labeled $\sim| p$. If S is a set of literals appearing in D, $D \mid\sim_{NDL} S$ if and only if for all $p \in S$, $D \mid\sim_{NDL} p$. $D \sim|_{NDL} S$ iff for some $p \in S$, $D \sim|_{NDL} p$.

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

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

Theorem 2 (Consistency): *If $D = \langle R, C, \prec\rangle$ is a defeasible theory, C is closed under strict rules, $S \in C$, and $D \mid\sim_{NDL} S$, then $\langle\{A \rightarrow p : A \rightarrow p \in R\}, C, \prec\rangle \mid\sim_{NDL} S$.*

Theorem 3 (Cautious Monotony): *If $D = \langle R, C, \prec\rangle$ is a defeasible theory, $D \mid\sim_{NDL} p$, and $D \mid\sim_{NDL} q$, then $\langle R \cup \{\rightarrow p\}, C, \prec\rangle \mid\sim_{NDL} q$.*

Theorem 1 assures us that we cannot both prove and demonstrate the absence of any proof for the same literal. Theorem 2 says that when conflict sets are closed under strict rules, any incompatible set of literals in NDL derivable from a defeasible theory must be derivable from the strict rules alone. In other words, the defeasible rules of a theory can never introduce any new incompatibilities. 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 the simple defeasible theory D_1 below.

Example 1

R_{D_1}: 1) $\rightarrow mammal$
2) $\{furry, has\_wings\} \Rightarrow bat$
3) $bat \Rightarrow furry$
4) $bat \Rightarrow has\_wings$
5) $bat \Rightarrow flies$
6) $mammal \Rightarrow \neg flies$

C_{D_1}: $\{bat, \neg bat\}$
$\{furry, \neg furry\}$
$\{has\_wings, \neg has\_wings\}$
$\{mammal, \neg mammal\}$
$\{flies, \neg flies\}$

$\prec_{D_1}$: $\emptyset$

In earlier versions of defeasible logic that lacked failure-by-looping, although we could easily see that there was no way to show $D_1 \mid\sim bat$, we could not demonstrate this in the proof theory, that is, we could not show $D_1 \sim| bat$. Consequently, neither could we show $D_1 \mid\sim \neg flies$. Failure-by-looping provides a mechanism for showing $D_1 \sim|_{NDL} bat$, which then allows us to show $D_1 \mid\sim_{NDL} \neg flies$.

When we later define a translation of defeasible theories into standard logic programs in such a way that the consequences of a theory correspond to the well-founded model 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_1 above is translated into Π_{D_1}, the literals bat, $furry$, and has_wings are all unfounded, but $\neg flies$ is well-founded. The corresponding literals 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 defeasible theory

Example 2

$$D_2 = \langle \{\Rightarrow p, \Rightarrow q, p \Rightarrow r, q \Rightarrow r, p \rightarrow \neg q, q \rightarrow \neg p\}, C, \emptyset \rangle$$

where C is closed under the strict rules in R. The corresponding default theory is

$$R_2 = \langle \{\neg(p \wedge q)\}, \{\tfrac{:p}{p}, \tfrac{:q}{q}, \tfrac{p:r}{r}, \tfrac{q:r}{r}\} \rangle$$

The classical part of R_2 containing the sentence $\neg(p \wedge q)$, acts like the conflict set in the defeasible theory and ensures that p and q do not both belong to the same extension. R_2 has two extensions containing $\{p, r\}$ and $\{q, r\}$, and the intersection of these extensions contains $\{r\}$. r is a "floating conclusion" of R_2 because it belongs to every extension even though there is no sentence that supports it that belongs to every extension.

This seems unintuitive to us. In NDL, the rules $\Rightarrow p$ and $\Rightarrow q$ conflict with each other since $\{p, q\}$ is a conflict set in the theory. Neither rule takes precedence over the other; so neither consequent is defeasibly derivable. Thus neither of the rules to establish r is satisfied, and r is also not defeasibly derivable. Our proof theory avoids these floating conclusions in an intuitively reasonable way.

2.1 Ambiguity Propagation and ADL

Consider the defeasible theory below in which conflict sets are minimal (from Brewka 2001).

Example 3

$$D_3 = \langle \{\Rightarrow p, \Rightarrow \neg p, \Rightarrow q, p \Rightarrow \neg q\}, C, \emptyset \rangle$$

We have $D_3 \sim\!|_{NDL}\ p$, $D_3 \sim\!|_{NDL}\ \neg p$, $D_3 \sim\!|_{NDL}\ \neg q$, and $D_3 \not\sim_{NDL}\ q$. This shows that NDL is *ambiguity blocking*. An argument exists for p, and hence $\neg q$, but this does not prevent us in NDL from concluding q. Many feel that *ambiguity propagating* systems, where such conclusions are forbidden, are intuitively more reasonable. An ambiguity propagating defeasible logic is presented in (Antoniou & Maher, 2002), and it is this variant that Brewka in (2001) considers for translation into logic programs. He dismisses an earlier, ambiguity blocking variant.

It turns out that a very minor modification to the proof system of NDL produces an ambiguity propagating defeasible logic (ADL), and it is precisely this modification that ensures that the results of a defeasible theory match those of its logic program counterpart under Brewka's translation (so that $D \sim_{ADL} p$ if and only if $p \in wfm(\Pi_D)$ and $D \sim\!|_{ADL}\ p$ if and only if $\neg p \in wfm(\Pi_D)$). The modification creating ADL affects only part 2.b.ii of Definition 6. It specifies that p is defeated only if every defeasible

rule in support of p fails or else is defeated by a satisfied strict rule or a defeasible rule of strictly *higher precedence* for each element $q \in cs - \{p\}$. In NDL, a rule of equal precedence can be used. The modification (Definition 8) is shown below.

Definition 8: τ is a defeasible proof in ADL iff τ is an argument tree for D, and for each node n of τ, one of the following obtains:

1. n is labeled $\mid\sim p$ and either:
   a. there is a strict rule $r : A \rightarrow p \in R$ such that A succeeds at n, or
   b. there is a defeasible rule $r : A \Rightarrow p \in R$ such that A succeeds at n and for all $cs \in C$, if $p \in cs$, then there is a $q \in cs - \{p\}$ such that for all rules $s : B \dashrightarrow q \in R$, either B fails at n or else $s \prec r$.
2. n is labeled $\sim\mid p$ and:
   a. for all strict rules $r : A \rightarrow p \in R$, A fails at n, and
   b. for all defeasible rules $r : A \Rightarrow p \in R$, either
      i. A fails at n, or
      ii. there is a $cs \in C$ such that $p \in cs$, and for all $q \in cs - \{p\}$, there is a rule $s : B \dashrightarrow q \in R$ such that B succeeds at n and s is strict or else $r \prec s$.
3. n is labeled $\sim\mid p$ and has an ancestor m labeled $\sim\mid p$, and all nodes between n and m are negative defeasible assertions.

Apart from this modification, all other aspects of the proof system are left alone. In example 3 above in ADL, one sees that since the rules for p and $\neg p$ are of the same precedence, neither $D_3 \sim\mid_{ADL} p$ nor $D_3 \sim\mid_{ADL} \neg p$ can be shown. Because $D_3 \sim\mid_{ADL} p$ cannot be shown, $D_3 \sim\mid_{ADL} \neg q$ cannot be shown, and so neither can $D_3 \mid\sim_{ADL} q$. Each of these literals is underdetermined in ADL, neither defeasibly proven nor refuted.

The default theory corresponding to D_3 is

$$R_3 = \langle \emptyset, \{ \tfrac{:p}{p}, \tfrac{:\neg p}{\neg p}, \tfrac{:q}{q}, \tfrac{p:\neg q}{\neg q} \} \rangle$$

R_3 has three default extensions containing $\{p, q\}$, $\{p, \neg q\}$, and $\{\neg p, q\}$. Notice that the intersection of these three extensions is "empty". So the "sceptical" interpretation of R_3 agrees with ADL in this case.

By examining the definition of proof trees for both NDL and ADL, it can be seen that every valid proof in ADL is a valid proof in NDL.

Theorem 4: *If D is a defeasible theory and $D \mid\sim_{ADL} p$, then $D \mid\sim_{NDL} p$. If $D \sim\mid_{ADL} p$, $D \sim\mid_{NDL} p$.*

ADL versions of Theorems 1, 2, and 3 also hold.

3 The Well-Founded Semantics for Normal Logic Programs

A logic program Π consists of a set of rules having the form

$$p \text{ :- } q_1, q_2, \ldots, q_n.$$

where p and each subgoal q_i is an atomic formula, or else such a formula preceded by the symbol not (often called negation-as-failure or negation-by-default). We will call a subgoal in which not occurs negative. The other subgoals are positive.

If a program contains no negative subgoals, then it is called *definite* and has a unique Herbrand model (Emden & Kowalski, 1976). This is often taken to be the intended meaning of the program. Programs in which not appears are called *normal* programs. Importantly, normal programs need not have a single least Herbrand model. E.g., the program $p :- not\ q$ has two minimal Herbrand models: $\{p\}$ and $\{q\}$. The *well-founded semantics* (Gelder, Ross, & Schlipf, 1988), (Gelder, Ross, & Schlipf, 1991) was developed to provide a reasonable interpretation of logic programs containing negation. For every program, a unique well-founded model exists. The well-founded model for the previous program is $\{p, \neg q\}$.

Let Π be a normal logic program containing only ground atoms and B_Π the set of atoms appearing in Π. An interpretation $\mathcal{I}$ of Π is any consistent set of positive and negative literals whose atoms are taken B_Π. 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 atoms $S \subseteq B_\Pi$ is said to be *unfounded* with respect to an interpretation $\mathcal{I}$ iff for each $p \in S$ and for each rule r of Π with head p, there exists a positive or negative subgoal q of r such that

1. q is false in $\mathcal{I}$, or
2. q is positive and $q \in S$.

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

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

$U_\Pi(\mathcal{I})$ can be viewed as a monotone operator and is used to derive the negative consequences of a program. The *immediate consequence operator* T, defined below, is used to derive the positive consequences of a program.

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

These two operators are combined to form a third:

$$W_\Pi(\mathcal{I}) = T_\Pi(\mathcal{I}) \cup \neg \cdot U_\Pi(\mathcal{I})$$

where $\neg \cdot U_\Pi(\mathcal{I})$ is the element-wise negation of $U_\Pi(\mathcal{I})$. $U_\Pi(\mathcal{I})$, $T_\Pi(\mathcal{I})$, and $W_\Pi(\mathcal{I})$ are all monotonic, and can be used to define the sequence

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

The well-founded model of Π, $wfm(\Pi)$, is the limit of this sequence.

4 Translating Defeasible Theories into Logic Programs

(Brewka, 2001) provides a simple and natural translation scheme to compare a version of defeasible logic (Antoniou & Maher, 2002) to logic programs using a prioritized well-founded semantics. Several examples are presented to demonstrate that the two systems do not agree, and it is argued there that the results of the defeasible theory are less reasonable.

We have altered the translation scheme to account for extended conflict sets and will use it to compare ADL to the WFS for normal programs. Only finite grounded defeasible theories and programs are considered. Since conflict sets are sufficient to encode negation, we will assume all literals appearing in a defeasible theory are positive ($\neg p$ is represented as p'). The translation thus necessarily yields a normal logic program. Furthermore, since (as discussed in the next section) defeaters and the precedence relation do not add to the expressiveness of defeasible logic, we will assume that no defeaters occur in the theory and that the precedence relation is empty.

Let $D = \langle R, C, \prec \rangle$ be a defeasible theory such that no defeaters occur in R and $\prec = \emptyset$. We define the logic program Π_D as follows.

1. If $q_1, q_2, \ldots, q_n \rightarrow p \in D$, then $p :- q_1, q_2, \ldots, q_n \in \Pi_D$.
2. Let cs_1, cs_2, ..., cs_m be the conflict sets of D containing p and $(a_1, a_2, \ldots, a_m)$ any tuple in $(cs_1 - \{p\}) \times (cs_2 - \{p\}) \times \ldots \times (cs_m - \{p\})$. If $q_1, q_2, \ldots, q_n \Rightarrow p \in D$, then $p :- not\ a_1, \ldots,\ not\ a_m, q_1, q_2, \ldots, q_n \in \Pi_D$.

Each a_i corresponds to some literal of a conflict set containing p, and in order for a defeasible rule to succeed at least one literal from each conflict set must fail. Significantly, if conflict sets are minimal, then the above translation of defeasible rules reduces to $p :- not\ p', q_1, q_2, \ldots, q_n$.

As the following examples show, the results of NDL are often incorrect with respect to the well-founded model of the Brewka inspired translation.

Example 4

$D_4 = \langle \{\Rightarrow p, \Rightarrow p'\}, \{p, p'\}, \emptyset \rangle$
$\Pi_{D_4} = \{p :- not\ p'., p' :- not\ p.\}$

Here, $D_4 \sim\!|_{NDL}\ p$ and $D_4 \sim\!|_{NDL}\ p'$ but the well founded model of Π_{D_4} is empty. According to the WFS, p is undefined.

Example 5

$D_5 = \langle \{p \rightarrow p\}, \{p, p'\}, \emptyset \rangle$
$\Pi_{D_5} = \{p :- p\}$

Both $D_5 \sim\!|_{NDL}\ p$ and $D_5 \sim\!|_{ADL}\ p$. Without failure-by-looping, neither of these results could be derived. p is unfounded in the well-founded model of Π_{D_5}.

Example 6

The logic program corresponding to D_3 is

$$\Pi_{D_3} = \{p :- \ not\ p'., p' :- \ not\ p.\ , q' :- \ not\ q, p., q :- \ not\ q'.\}$$

D_4 and D_5 together show in general that, for a defeasible theory D, $D \sim|_{NDL}\ p$ is equivalent to neither $\neg p \in wfm(\Pi_D)$ nor p undefined. In Example 6, we have $D_3 \sim|_{NDL}\ p$, $D_3 \sim|_{NDL}\ p'$, $D_3 \sim|_{NDL}\ q'$, and $D_3 \not\sim_{NDL}\ q$. NDL is *ambiguity blocking*. However, the well-founded model of the logic program is empty. In general, for a defeasible theory D, $D \not\sim_{NDL}\ p$ does not imply $p \in wfm(\Pi_D)$. In some cases at least, the well-founded semantics is more conservative than NDL. In contrast, the assertions of ADL are correct with respect to the well-founded semantics.

Theorem 5: *Let $D = \langle R, C, \emptyset \rangle$ be a defeasible theory of ADL without defeaters. Then $D \not\sim_{ADL}\ p$ if and only if $p \in wfm(\Pi_D)$, and $D \sim|_{ADL}\ p$ if and only if $\neg p \in wfm(\Pi_D)$.*

In Example 4, no superior argument exists for either p or p', and so none of $D_4 \sim|_{ADL}\ p$, $D_4 \sim|_{ADL}\ p'$, $D_4 \not\sim_{ADL}\ p$, or $D_4 \not\sim_{ADL}\ p'$ can be shown. In Example 5, in agreement with both NDL and the WFS, $D \sim|_{ADL}\ p$ can be shown using failure-by-looping. In Example 6, arguments exist for both p and p', but no superior argument exists for either. They are both ambiguous, and this ambiguity prevents concluding anything about q or q'.

5 Eliminating Precedence and Defeaters from Defeasible Logic

Theorem 5 assumes that the defeasible theory to be translated does not contain any defeaters and that the precedence relation is empty. However, eliminating these from defeasible logic does not lessen its expressiveness.

For any defeasible theory $D = \langle R_D, C_D, \prec_D \rangle$ we can construct another, E, lacking defeaters and with an empty precedence relation, such that D and E agree on all the literals appearing in D.

The conflict sets in E are minimal and defined using literals from R_E, and $\prec_E$ is empty. R_E is constructed by explicitly representing the rules of D, as described below.

1. If r: $A \rightarrow p \in D$, then r_a = $A \rightarrow supported(r)$, r_b = $supported(r) \rightarrow fires(r)$, and r_c = $fires(r) \rightarrow p$ appear in R_E.
2. If r: $A \Rightarrow p \in D$, then r_a = $A \rightarrow supported(r)$, r_b = $supported(r) \Rightarrow fires(r)$, r_c = $fires(r) \rightarrow p$, appear in R_E:
3. If r: $A \rightsquigarrow p \in D$, then r_a = $A \rightarrow supported(r)$ and r_b = $supported(r) \Rightarrow fires(r)$ appear in R_E.
4. Let $cs = \{q_1, q_2, \ldots q_n, p\}$ be a conflict set of D, r a defeasible or defeater rule with head p and $s_1, s_2, \ldots, s_n$ rules such that s_i has head q_i and $s_i \not\prec r$. If for each s_i, s_i is strict or $r \prec s_i$, the rule $supported(s_1), \ldots supported(s_n) \rightarrow \neg fires(r)$ occurs in E. Otherwise, the rule $supported(s_1), \ldots supported(s_n) \Rightarrow \neg fires(r)$ occurs in E.

Theorem 6: *Let D and E be defeasible theories as defined and p a literal of D. Then $D \not\sim_{ADL}\ p$ if and only if $E \not\sim_{ADL}\ p$, and $D \sim|_{ADL}\ p$ if and only if $E \sim|_{ADL}\ p$.*

We have added new literals of the form $fires(r)$ and $supported(r)$ which explicitly encode when a rule r of D may fire (i.e., the head is derivable) and when it is supported (ie., its body is derivable). Intuitively, if r is strict, then it may fire if and only if its body is supported. This is represented in r_a, r_b, and r_c, all of which are strict. If r is defeasible or a defeater, r_b is defeasible, meaning that a rule in E with head $\neg fires(r)$ can potentially defeat it. If r is a defeater, r_a and r_b bear no relation to p and cannot be used to derive it.

In the item 4 above, we have explicitly represented when a set of rules $s_1, \ldots, s_m$ can be used to defeat another r. We need only consider an s_i if it is not inferior to r, for inferior rules cannot be used to defeat r.

6 Translating Normal Logic Programs into Defeasible Logic

Here we show that normal logic programs can also be translated into defeasible theories so that the assertions of the theory match the well-founded model. Let Π be a normal program and B_Π the set of atoms in Π. For each atom $b \in B_\Pi$, define b' to be a new atom not appearing in Π. Given these, we define a new program Φ as follows.

1. If $p :- a_1, \ldots, a_n,\ not\ b_1, \ldots,\ not\ b_m \in \Pi$, then $p :- a_1, \ldots, a_n, b'_1, \ldots, b'_m \in \Phi$.
2. For all b'_i, $b'_i :- not\ b_i \in \Phi$.

As each new atom b' has exactly one rule, and the only subgoal of that rule is $not\ b$, b' occurs in the well-founded model of Φ if and only if $\neg b$ does.

Lemma 1: *Let Π and Φ be programs as defined above. For any $b_i \in B_\Pi$, $b'_i \in wfm(\Phi)$ iff $\neg b_i \in wfm(\Phi)$. Also, $\neg b'_i \in wfm(\Phi)$ iff $b_i \in wfm(\Phi)$.*

We may view b' as the positive representation of '$\neg b$' (and for normal programs it is impossible for b' and b to both be in the well-founded model). Furthermore, with respect to the original atoms of B_Π, Π and Φ are equivalent under the WFS.

Lemma 2: *Let Π and Φ be programs as defined above. For all $p \in B_\Pi$, $p \in wfm(\Pi)$ iff $p \in wfm(\Phi)$, and $\neg p \in wfm(\Pi)$ iff $\neg p \in wfm(\Phi)$.*

The relationship between Π and Φ makes a translation of Π into a defeasible theory D_Π apparent. Let $r = p :- a_1, \ldots a_n,\ not\ b_1, \ldots,\ not\ b_m$ be a rule of Π. Define r_{D_Π} to be $a_1, \ldots a_n, b'_1 \ldots, b'_m \rightarrow p$. Let $Str = \{r_{D_\Pi} |\ r \in \Pi\}$, and $Pr = \{\emptyset \Rightarrow p' |\ not\ p$ occurs in some rule of $\Pi\}$. Given this, D_Π is defined as follows:

$$R_{D_\Pi} = \langle Str \cup Pr, C, \emptyset \rangle$$

where C is minimal. The default literals in the program have become presumptions in the defeasible theory. The rules of the original program are strict in the defeasible logic theory.

It is clear given the above definition that translating D_Π into a logic program yields Φ. From Lemma 1 and Theorem 5 it follows that p is provable in D_Π if and only if p' is

refutable, and p' is provable if and only if p is refutable. Furthermore, since according to Theorem 5 the assertions of D_Π correspond to the well-founded model of Φ, by Lemma 2 the assertions of D_Π agree with the well-founded model of Π wrt B_Π.

Theorem 7: *Let D_Π be defined as above. For any $p \in B_\Pi$, $D_\Pi \mid\sim_{ADL} p$ iff $D_\Pi \sim|_{ADL} p'$, and $D_\Pi \mid\sim_{ADL} p'$ iff $D_\Pi \sim|_{ADL} p$.*

Theorem 8: *Let Π be a normal program and D_Π its defeasible logic translation. For any atom $p \in B_\Pi$, $D_\Pi \mid\sim_{ADL} p$ iff $p \in wfm(\Pi)$, and $D_\Pi \sim|_{ADL} p$ iff $\neg p \in wfm(\Pi)$.*

Example 7

$$
\begin{array}{lllll}
\Pi = & \{p :- not\ q, & q :- not\ p\} & & \\
\Phi = & \{p :- q', & q :- p', & p' :- not\ p, & q' :- not\ q\} \\
D_\Pi = & \langle\{q' \rightarrow p, & p' \rightarrow q, & \Rightarrow p', \Rightarrow q'\}, & C, \emptyset\rangle
\end{array}
$$

Here we have replaced in the rules of Π each subgoal $not\ p$ (alternatively $not\ q$) with p' (alternatively q') and added the rules $p' :- not\ p$ and $q' :- not\ q$. The explicitly negative literals p' and q' occur nowhere in the original program Π and so it is safe to equate, e.g., $not\ p$ with p'. The well-founded model of both Π and Φ is empty. In D_Π, the presumption of p' prevents concluding q', but there is no superior argument for q, and so q' is not refuted, either. The same holds for p', and because of this nothing can be determined for p or q. The set of assertions of D_Π is empty.

In an extended logic program, an explicitly negative literal p' might already appear in Π, and so the manoeuver of replacing $not\ p$ with p' is no longer acceptable (the equivalence of Φ and Π would no longer hold. For instance, if $\Pi = \{p :- \emptyset, p' :- \emptyset, q :- not\ p\}$, then the well-founded model is $\{p, p', \neg q\}$. However, $\Phi = \{p :- \emptyset, p' :- \emptyset, q :- p', p' :- not\ p\}$, and the well-founded model of Φ is $\{p, p', q\}$.

If we instead replace $not\ p$ with some entirely new literal c_p, then the equivalence is restored. However, in the defeasible logic translation, p and p' do not conflict (the program rule $c_p :- not\ p$ would be interpreted as a $\emptyset \Rightarrow c_p$ and c_p and p would conflict). This at first seems very odd. However, strictly speaking, there really is no connection between p and p' in the WFS, either. The literal p' is just a rather strange looking atom.

7 Conclusion

The two versions of defeasible logic presented here differ in that one, NDL, blocks ambiguity while the other, ADL, does not. In a previous paper (Maier and Nute, 2006) we showed how to translate finite defeasible theories into normal logic programs in such a way that the consequences of the theory in NDL correspond to the well-founded model of the logic program. In this paper, we showed a similar result for ADL. Furthermore, we showed in this paper how to translate finite normal logic programs into defeasible theories in such a way that the correspondence between the well-founded model of the program and the consequences of the defeasible theory in ADL are preserved. The

correspondence between logic programs and defeasible theories seems to depend on the fact that ADL is ambiguity propagating rather than ambiguity blocking. The translation from defeasible theories to normal logic programs seems to us simpler and more direct for the case of ADL than NDL. Some might consider this a reason to prefer ADL over NDL. But different researchers have taken different positions on this issue, and our intuitions favor ambiguity blocking despite the closer correspondence of ADL with well-founded semantics.

In Example 3, the two presumptions $\Rightarrow p$ and $\Rightarrow \neg p$ defeat each other, but in ADL $\Rightarrow p$ and $p \Rightarrow \neg q$ are still available to defeat the presumption $\Rightarrow q$, creating a "zombie path" (Makinson & Schechta, 1991), an argument path that has been "killed" by a defeater but that still has the power to defeat or "kill" other arguments. So we get $D_1 \mid\!\sim_{NDL} p$ and $D_1 \sim\!|_{NDL} \{p, \neg p, \neg q\}$, while we get $D_1 \mid\!\sim_{ADL} \emptyset$ and $D_1 \sim\!|_{ADL} \emptyset$. In the corresponding default theory, the intersection of all extensions was empty, agreeing with ADL. Nevertheless, we think that lacking overriding reasons for accepting either p or $\neg p$, we should take neither into account when considering q, the approach taken in NDL.

So far as positive consequences are concerned, Example 2 is handled in the same way by both NDL and ADL: we get neither $D_2 \mid\!\sim_{NDL} r$ nor $D_2 \mid\!\sim_{ADL} r$. In both systems, the two presumptions $\Rightarrow p$ and $\Rightarrow q$ conflict because $\{p, q\}$ is a conflict set in D_2. We believe this is the correct result in examples of this sort. However, we get $D_2 \sim\!|_{NDL} \{p, q, r\}$, but $D_2 \sim\!|_{ADL} \emptyset$.

Our results for defeasible theories and logic programs assume that the theories and programs are finite, but they should also hold for theories that do not have infinite chains of dependency. In a theory like

$$D = \langle \{\Rightarrow p, q_1 \Rightarrow \neg p\} \cup \{q_{i+1} \Rightarrow q_i : i \text{ a positive integer }\}, C, \emptyset \rangle$$

our defeasible logics and well-founded semantics for logic programs diverge. Since proof trees in either NDL or ADL are finite, not even failure-by-looping will allow us to show that p is derivable in either system. However, p will be in the well-founded model for the corresponding logic program. Suppose we admit semi-infinite proofs in NDL and ADL. A semi-infinite proof would be a tree all of whose infinite branches have a node n such that all descendants of n in the infinite branch are labeled with negative defeasible assertions. We think that no additional conditions are needed for semi-infinite proofs, but we have not investigated this problem further. The question whether the consequences of an infinite defeasible theory in such a system corresponds to the well-founded model for the corresponding infinite logic program is interesting in principal, but finite, constructive proof procedures for such theories and logic programs would in the general case not exist.

Translating defeasible theories into logic programs offers one way to implement NDL and ADL. Whether this or some more direct method for computing consequences is the better method depends in large part on the cost of translating defeasible theories into logic programs. We have not done a complete analysis of the complexity issues yet, but at first glance it appears that when conflict sets are closed under strict rules, translating a defeasible theory into a logic program might require more than polynomial time. This might not be so bad if the translation only had to be performed once or if the

translation procedure were modular. If new facts are added to a defeasible theory, then these facts can be translated into an already existing program in linear time. But adding new rules, and particularly new defeasible rules, looks like it will also require more than polynomial time. So in applications where new rules will not be added very often, then translation into a logic program seems to be a reasonable approach. The more often new rules, including new presumptions, are added to a theory, the less attractive this approach appears.

References

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

Antoniou, G.; Billington, D.; Governatori, G.; Maher, M. J.; and Rock, A. 2000. A family of defeasible reasoning logics and its implementation. In *ECAI*, 459–463.

Brewka, G. 2001. On the relationship between defeasible logic and well-founded semantics. In *LPNMR*, 121–132.

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.

Maier, F., and Nute, D. 2006. Relating defeasible logic to the well-founded semantics for normal logic programs. In Delgrande, J. P., and Schaub, T., eds., *NMR*.

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.