

A Semantic Decomposition of Defeasible Logics

M.J. Maher and G. Governatori

School of Computing and Information Technology, Griffith University
Nathan, QLD 4111, Australia
{mjm, guido}@cit.gu.edu.au

Abstract

We investigate defeasible logics using a technique which decomposes the semantics of such logics into two parts: a specification of the structure of defeasible reasoning and a semantics for the meta-language in which the specification is written. We show that Nute's Defeasible Logic corresponds to Kunen's semantics, and develop a defeasible logic from the well-founded semantics of Van Gelder, Ross and Schlipf. We also obtain a new defeasible logic which extends an existing language by modifying the specification of Defeasible Logic. Thus our approach is productive in analysing, comparing and designing defeasible logics.

Introduction

In this paper we start from Nute's Defeasible Logic (Nute, 1987; Nute 1994). This logic has an expressive syntax, a strongly skeptical semantics and a tractable computational behavior. Our interest, in this paper, is to decompose Defeasible Logic into parts, for the analysis of the logic, and also to reassemble it with different parts to create new and different logics.

We show that one component of Defeasible Logic is Kunen's semantics for logic programs (Kunen, 1987). As a consequence of this link, inference in predicate Defeasible Logic (where arbitrary function symbols are allowed) is computable, and inference in propositional Defeasible Logic is polynomial.

The technique that we use – meta-programming – allows us to provide several different semantics to the syntactic elements of Defeasible Logic without violating the underlying intuitive meaning of the syntax. Thus we can create several different defeasible logics, all adhering to the defeasible structure underlying Defeasible Logic. (Of course, the computational complexity of such logics varies with the semantics.) In particular, we show that a defeasible logic developed using unfounded sets corresponds exactly to the use of the well-founded semantics of logic programs (Van Gelder et al., 1991).

The paper is organized as follows. The next section introduces Defeasible Logic and its proof theory. We establish a bottom-up characterization of the consequences of a defeasible theory that serves as our semantics for Defeasible

Logic. We also define Well-Founded Defeasible Logic and show that it is coherent and consistent.

In the third section we present the metaprogram that encodes the basic behavior of the Defeasible Logic syntactic constructs. We outline Kunen's semantics and the well-founded semantics of logic programs, and show that the composition of these semantics with the metaprogram produces, respectively, Defeasible Logic and Well-Founded Defeasible Logic. We also show, using the metaprogram, that explicit failure operators can be added to defeasible logics in a conservative way. Finally, we present some future work and conclusions.

Defeasible Logic

Outline of Defeasible Logic

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* $C(r)$ which is a literal. In writing rules we omit set notation for antecedents.

There are three kinds of rules: *Strict rules* are denoted by $A \rightarrow p$, and are interpreted in the classical sense: whenever the premises are indisputable (e.g. facts) then so is the conclusion. An example of a strict rule is "Emus are birds". Written formally: $emu(X) \rightarrow bird(X)$. Inference from facts and strict rules only is called *definite inference*.

Defeasible rules are denoted by $A \Rightarrow p$, and can be defeated by contrary evidence. An example of such a rule is $bird(X) \Rightarrow flies(X)$, which reads as follows: "Birds typically fly".

Defeaters are denoted by $A \rightsquigarrow p$ and are used to prevent some conclusions. In other words, they are used to defeat some defeasible rules by producing evidence to the contrary. An example is the rule $heavy(X) \rightsquigarrow \neg flies(X)$, which reads as follows: "If an animal is heavy then it may not be able to fly". The main point is that the information that an animal is heavy is not sufficient evidence to conclude that it doesn't fly. It is only evidence that the animal *may* not be able to fly.

A *superiority relation on R* is a relation $>$ on R (that is, the transitive closure of $>$ is irreflexive). When $r_1 > r_2$, then r_1 is called *superior* to r_2 , and r_2 *inferior* to r_1 . This expresses that r_1 may override r_2 . For example, given the defeasible rules

$$\begin{aligned} r : \quad & \text{bird}(X) \Rightarrow \text{flies}(X) \\ r' : \quad & \text{brokenWing}(X) \Rightarrow \neg \text{flies}(X) \end{aligned}$$

which contradict one another, no conclusive decision can be made about whether a bird with a broken wing can fly. But if we introduce a superiority relation $>$ with $r' > r$, then we can indeed conclude that it cannot fly.

A defeasible theory consists of a set of facts, a set of rules, and a superiority relation. A *conclusion* of a defeasible theory D 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 D .
- $-\Delta q$, which is intended to mean that we have proved that q is not definitely provable in D .
- $+\partial q$, which is intended to mean that q is defeasibly provable in D .
- $-\partial q$ which is intended to mean that we have proved that q is not defeasibly provable in D .

Definite provability involves only strict rules and facts.

Proof Theory

In this presentation we use the formulation of Defeasible Logic given in (Billington 1993). A *defeasible theory* D is a triple $(F, R, >)$ where F is a set of literals (called *facts*), R a finite set of rules, and $>$ a superiority relation on R . In expressing the proof theory we consider only propositional rules. Rules such as the previous examples are interpreted as the set of their variable-free instances.

Given a set R of rules, we denote the set of all strict rules in R by R_s , the set of strict and defeasible rules in R by R_{sd} , the set of defeasible rules in R by R_d , and the set of defeaters in R by R_{dft} . $R[q]$ denotes the set of rules in R with consequent q . In the following $\sim p$ denotes the complement of p , that is, $\sim p$ is $\neg p$ if p is an atom, and $\sim p$ is q if p is $\neg q$.

Provability is defined below. It is based on the concept of a *derivation* (or *proof*) in $D = (F, R, >)$. A derivation is a finite sequence $P = (P(1), \dots, P(n))$ of tagged literals satisfying the following conditions ($P(1..i)$ denotes the initial part of the sequence P of length i):

$$\begin{aligned} +\Delta: \text{ If } P(i+1) = +\Delta q \text{ then either} \\ & q \in F \text{ or} \\ & \exists r \in R_s[q] \forall a \in A(r) : +\Delta a \in P(1..i) \end{aligned}$$

That means, to prove $+\Delta q$ we need to establish a proof for q using facts and strict rules only. This is a deduction in the classical sense – no proofs for the negation of q need to be considered (in contrast to defeasible provability below, where opposing chains of reasoning must be taken into account, too).

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

To prove $-\Delta q$, i.e. that q is not definitely provable, q must not be a fact. In addition, we need to establish that every strict rule with head q is *known to be* inapplicable. Thus for every such rule r there must be at least one antecedent a for which we have established that a is not definitely provable ($-\Delta a$).

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

Let us illustrate this definition. To show that q is provable defeasibly we have two choices: (1) We show that q is already definitely provable; or (2) we need to argue using the defeasible part of D as well. In particular, we require that there must be a strict or defeasible rule with head q which can be applied (2.1). But now we need to consider possible “counterattacks”, that is, reasoning chains in support of $\sim q$. To be more specific: to prove q defeasibly we must show that $\sim q$ is not definitely provable (2.2). Also (2.3) we must consider the set of all rules which are not known to be inapplicable and which have head $\sim q$ (note that here we consider defeaters, too, whereas they could not be used to support the conclusion q ; this is in line with the motivation of defeaters given above). Essentially each such rule s attacks the conclusion q . For q to be provable, each such rule s must be counterattacked by a rule t with head q with the following properties: (i) t must be applicable at this point, and (ii) t must be stronger than (i.e. superior to) s . Thus each attack on the conclusion q must be counterattacked by a stronger rule.

The definition of the proof theory of defeasible logic is completed by the condition $-\partial$. It is nothing more than a strong negation of the condition $+\partial$.

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

To prove that q is not defeasibly provable, we must first establish that it is not definitely provable. Then we must establish that it cannot be proven using the defeasible part of the theory. There are three possibilities to achieve this: either we have established that none of the (strict and defeasible) rules with head q can be applied (2.1); or $\sim q$ is definitely provable (2.2); or there must be an applicable rule s with head $\sim q$ such that no applicable rule t with head q is superior to s .

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

Under some assumptions, the logic and conditions concerning defeasible provability can be simplified (Antoniou et al., 1998) but we do not need those assumptions here.

A Bottom-Up Characterization of Defeasible Logic

The proof theory provides the basis for a top-down (backward-chaining) implementation of the logic. However, there are advantages to a bottom-up (forward-chaining) implementation. Furthermore, a bottom-up definition of the logic provides a bridge to the logics we will define later. For these reasons we now provide a bottom-up definition of Defeasible Logic.

We associate with D an operator \mathcal{T}_D which works on 4-tuples of sets of literals. We call such 4-tuples an *extension*.

$$\mathcal{T}_D(+\Delta, -\Delta, +\partial, -\partial) = (+\Delta', -\Delta', +\partial', -\partial') \text{ where}$$

$$+\Delta' = F \cup \{q \mid \exists r \in R_s[q] A(r) \subseteq +\Delta\}$$

$$-\Delta' = -\Delta \cup (\{q \mid \forall r \in R_s[q] A(r) \cap -\Delta \neq \emptyset\} - F)$$

$$+\partial' = +\Delta \cup \{q \mid \exists r \in R_{sd}[q] A(r) \subseteq +\partial, \\ \sim q \in -\Delta, \text{ and} \\ \forall s \in R[\sim q] \text{ either} \\ A(s) \cap -\partial \neq \emptyset, \text{ or} \\ \exists t \in R[q] \text{ such that} \\ A(t) \subseteq +\partial \text{ and } t > s\}$$

$$-\partial' = \{q \in -\Delta \mid \forall r \in R_{sd}[q] A(r) \cap -\partial \neq \emptyset, \text{ or} \\ \sim q \in +\Delta, \text{ or} \\ \exists s \in R[\sim q] \text{ such that } A(s) \subseteq +\partial \text{ and} \\ \forall t \in R[q] \text{ either} \\ A(t) \cap -\partial \neq \emptyset, \text{ or} \\ t \not> s\}$$

The set of extensions forms a complete lattice under the pointwise containment ordering¹, with $\perp = (\emptyset, \emptyset, \emptyset, \emptyset)$ as its least element. The least upper bound operation is the pointwise union, which is represented by \cup . It can be shown that \mathcal{T}_D is monotonic and the Kleene sequence from \perp is increasing. Thus the limit $F = (+\Delta_F, -\Delta_F, +\partial_F, -\partial_F)$ of all finite elements in the sequence exists, and \mathcal{T}_D has a least fixpoint $L = (+\Delta_L, -\Delta_L, +\partial_L, -\partial_L)$. When D is a finite propositional defeasible theory $F = L$.

The extension F captures exactly the inferences described in the proof theory.

Theorem 1 *Let D be a finite propositional defeasible theory and q a literal.*

- $D \vdash +\Delta q$ iff $q \in +\Delta_F$
- $D \vdash -\Delta q$ iff $q \in -\Delta_F$
- $D \vdash +\partial q$ iff $q \in +\partial_F$
- $D \vdash -\partial q$ iff $q \in -\partial_F$

The restriction of Theorem 1 to finite propositional theories derives from the formulation of the proof theory; proofs are guaranteed to be finite under this restriction. However, the bottom-up semantics and the following work do

¹ $(a_1, a_2, a_3, a_4) \leq (b_1, b_2, b_3, b_4)$ iff $a_i \subseteq b_i$ for $i = 1, 2, 3, 4$.

not need this restriction, and so apply to predicate defeasible logic rules that represent infinitely many propositional rules. Indeed, for the remainder of this paper we will take the bottom-up semantics F as representative of Defeasible Logic in this wider sense. We will write $D \vdash +\Delta q$ to express $q \in +\Delta_F$, and similarly with other conclusions.

The analysis of the proof theory of Defeasible Logic in (Maher et al., 1998) was based on a 4-tuple defined purely in terms of the (top-down) proof theory. By Theorem 1, that 4-tuple is precisely F .

An extension $(+\Delta, -\Delta, +\partial, -\partial)$ is *coherent* if $+\Delta \cap -\Delta = \emptyset$ and $+\partial \cap -\partial = \emptyset$. An extension is *consistent* if whenever $p \in +\partial$ and $\sim p \in +\partial$, for some p , then also $p \in +\Delta$ and $\sim p \in +\Delta$. Intuitively, coherence says that no literal is simultaneously provable and unprovable. Consistency says that a literal and its negation can both be defeasibly provable only when it and its negation are definitely provable; hence defeasible inference does not introduce inconsistency. A logic is coherent (consistent) if the meaning of each theory of the logic, when expressed as an extension, is coherent (consistent).

The following result was shown in (Billington, 1993).

Proposition 2 *Defeasible Logic is coherent and consistent.*

Well-Founded Defeasible Logic

It follows from the above definitions that defeasible theories such as

$$r : p \Rightarrow p$$

conclude neither $+\partial p$ nor $-\partial p$. In some contexts it is desirable for a logic to recognize such “loops” and to conclude $-\partial p$. Building on the bottom-up definition of the previous subsection, and inspired by the work of (Van Gelder et al., 1991), we define a well-founded defeasible logic which draws such conclusions.

The central definition required is that of an unfounded set. Since defeasible logic involves both definite and defeasible inference, we need two definitions. A set S of literals is unfounded with respect to an extension E and definite inference (or Δ -unfounded) if: For every literal s in S , and for every strict rule $B \rightarrow s$ either

- $B \cap -\Delta_E \neq \emptyset$, or
- $B \cap S \neq \emptyset$

This definition is very similar to the definition of unfounded set in (Van Gelder et al., 1991). The main differences are that the basic elements of S are literals (and negation is classical negation) and “negation as failure” is not present in the bodies of rules.

The corresponding definition for defeasible inference is more complex, since there are more factors that influence defeasible inference. Nevertheless, the basic idea is the same.

We use \hookrightarrow to denote that the arrow of a rule is not specified. That is, $B \hookrightarrow s$ refers to a rule that might be strict, defeasible, or a defeater.

A set S of literals is unfounded with respect to an extension E and defeasible inference (or ∂ -unfounded) if: For every literal s in S , and for every strict or defeasible rule $r_1 : A(r_1) \hookrightarrow s$ in D either

- $A(r_1) \cap -\partial_E \neq \emptyset$, or
- $A(r_1) \cap S \neq \emptyset$, or
- there is a rule $r_2 : A(r_2) \hookrightarrow \sim s$ in D such that $A(r_2) \subseteq +\partial_E$ and for every rule $r_3 : A(r_3) \hookrightarrow s$ in D either
 - $A(r_3) \cap -\partial_E \neq \emptyset$, or
 - $r_3 \not\prec r_2$.

Clearly the classes of Δ -unfounded and ∂ -unfounded sets are both closed under unions. Hence there is a greatest Δ -unfounded set wrt E (denoted by $U_D^\Delta(E)$), and a greatest ∂ -unfounded set wrt E (denoted by $U_D^\partial(E)$). Let $\mathcal{U}_D(E) = (\emptyset, U_D^\Delta(E), \emptyset, U_D^\partial(E))$.

We define $\mathcal{W}_D(E) = \mathcal{T}_D(E) \cup \mathcal{U}_D(E)$.

Let I_α be the elements of the (possibly transfinite) Kleene sequence starting from $\perp = (\emptyset, \emptyset, \emptyset, \emptyset)$. $\{I_\alpha \mid \alpha \geq 0\}$ is an increasing sequence and thus has a limit.

Let $WF = (+\Delta_{WF}, -\Delta_{WF}, +\partial_{WF}, -\partial_{WF})$ be the limit of this sequence. Then WF defines the conclusions of Well-Founded Defeasible Logic. If a literal $q \in +\Delta_{WF}$ we write $D \vdash_{WF} +\Delta q$, and similarly with the three other sets in WF .

We can verify that the resulting logic is sensible in the following sense.

Proposition 3 *Well-Founded Defeasible Logic is coherent and consistent.*

To illustrate the definitions, consider the following Well-Founded Defeasible Logic theory.

$$\begin{array}{ll}
r_1 : & b \Rightarrow a \\
r_2 : & \neg c \Rightarrow a \\
r_3 : & d \Rightarrow a \\
r_4 : & a \Rightarrow \neg c \\
r_5 : & true \Rightarrow d \\
r_6 : & true \Rightarrow \neg d
\end{array}$$

With respect to the extension E where $-\partial a, -\partial b$ and $-\partial c$ hold, an unfounded set is $U = \{a, \neg c, d, \neg d\}$. Well-Founded Defeasible Logic will conclude $-\partial a, -\partial \neg c, -\partial d$ and $-\partial \neg d$, whereas conventional Defeasible Logic will conclude only $-\partial d$ and $-\partial \neg d$.

Decomposition of Defeasible Logics

In this section we show how a defeasible logic can be decomposed into a metaprogram specifying the structure of defeasible reasoning, and a semantics for the meta-language (logic programming). We first introduce the metaprogram, then the two semantics that, when composed with the metaprogram, produce the two logics defined previously. Finally we discuss an example where the metaprogram is modified.

The Defeasible Logic Metaprogram

In this section we introduce a metaprogram \mathcal{M} in a logic programming form that expresses the essence of the defeasible reasoning embedded in the proof theory. The metaprogram assumes that the following predicates, which are used to represent a defeasible theory, are defined.

- $\text{fact}(Head)$,

- $\text{strict}(Name, Head, Body)$,
- $\text{defeasible}(Name, Head, Body)$,
- $\text{defeater}(Name, Head, Body)$, and
- $\text{sup}(Rule1, Rule2)$,

\mathcal{M} consists of the following clauses. We first introduce the predicates defining classes of rules, namely

```

supportive_rule(Name, Head, Body):-
  strict(Name, Head, Body).

supportive_rule(Name, Head, Body):-
  defeasible(Name, Head, Body).

rule(Name, Head, Body):-
  supportive_rule(Name, Head, Body).

rule(Name, Head, Body):-
  defeater(Name, Head, Body).

```

We introduce now the clauses defining the predicates corresponding to $+\Delta$, $-\Delta$, $+\partial$, and $-\partial$. These clauses specify the structure of defeasible reasoning in Defeasible Logic. Arguably they convey the conceptual simplicity of Defeasible Logic more clearly than does the proof theory.

```

c1  definitely(X):-
    fact(X).

c2  definitely(X):-
    strict(R, X, [Y1, ..., Yn]),
    definitely(Y1), ..., definitely(Yn).

c3  not_definitely(X):-
    not definitely(X).

c4  defeasibly(X):-
    definitely(X).

c5  defeasibly(X):-
    not definitely(~ X),
    supportive_rule(R, X, [Y1, ..., Yn]),
    defeasibly(Y1), ..., defeasibly(Yn),
    not overruled(S, R, X).

c6  overruled(S, R, X):-
    sup(S, R),
    rule(S, ~ X, [U1, ..., Un]),
    defeasibly(U1), ..., defeasibly(Un),
    not defeated(T, S, ~ X).

c7  defeated(T, S, ~ X):-
    sup(T, S),
    supportive_rule(T, X, [V1, ..., Vn]),
    defeasibly(V1), ..., defeasibly(Vn).

c8  not_defeasibly(X):-
    not defeasibly(X).

```

The first three clauses address definite provability, while the remainder address defeasible provability. The clauses specify if and how a rule in Defeasible Logic can be overridden by another, and which rules can be used to defeat an over-riding rule, among other aspects of the structure of defeasible reasoning.

We have permitted ourselves some syntactic flexibility in presenting the metaprogram. However, there is no technical

difficulty in using conventional logic programming syntax to represent this program.

This metaprogram is similar to – though briefer and more intelligible than – the meta-interpreter d-Prolog for Defeasible Logic defined in (Covington et al., 1997). The d-Prolog meta-interpreter was designed for execution by Prolog, with the Prolog implementation of negation-as-failure. It contains many complications due to this intended use.

Given a defeasible theory $D = (F, R, >)$, the corresponding program \mathcal{D} is obtained from \mathcal{M} by adding facts according to the following guidelines:

1. `fact(p)`. for each $p \in F$;
2. `strict($r_i, p, [q_1, \dots, q_n]$)`.
for each rule $r_i : q_1, \dots, q_n \rightarrow p \in R$;
3. `defeasible($r_i, p, [q_1, \dots, q_n]$)`.
for each rule $r_i : q_1, \dots, q_n \Rightarrow p \in R$;
4. `defeater($r_i, p, [q_1, \dots, q_n]$)`.
for each rule $r_i : q_1, \dots, q_n \rightsquigarrow p \in R$;
5. `sup(r_i, r_j)`.
for each pair of rules such that $r_i > r_j$.

Kunen Semantics

Kunen’s semantics (Kunen, 1987) is a 3-valued semantics for logic programs. A *partial interpretation* is a mapping from ground atoms to one of three truth values: **t** (true), **f** (false), and **u** (unknown). This mapping can be extended to all formulas using Kleene’s 3-valued logic.

Kleene’s truth tables can be summarized as follows. If ϕ is a boolean combination of the atoms **t**, **f**, and **u**, its truth value is **t** iff all the possible ways of putting in **t** or **f** for the various occurrences of **u** lead to a value **t** being computed in ordinary 2-valued logic: ϕ gets the value **f** iff $\neg\phi$ gets the value **f**, and ϕ gets the value **u** otherwise. These truth values can be extended in the obvious way to predicate logic, thinking of the quantifiers as infinite disjunction or conjunction.

The Kunen semantics of a program \mathcal{P} is obtained from a sequence $\{I_n\}$ of partial interpretations, defined as follows.

1. $I_0(\alpha) = \mathbf{u}$ for every atom α
2. $I_{n+1}(\alpha) = \mathbf{t}$ iff for some clause

$$\beta:-\phi$$

in the program, $\alpha = \beta\sigma$ for some ground substitution σ such that

$$I_n(\phi\sigma) = \mathbf{t} .$$

3. $I_{n+1}(\alpha) = \mathbf{f}$ iff for all the clauses

$$\beta:-\phi$$

in the program, and all ground substitution σ , if $\alpha = \beta\sigma$, then

$$I_n(\phi\sigma) = \mathbf{f} .$$

4. $I_{n+1}(\alpha) = \mathbf{u}$ otherwise.

We shall say that the Kunen semantics of \mathcal{P} supports α iff there is an interpretation I_n , for some finite n , such that

$I_n(\alpha) = \mathbf{t}$. This semantics has an equivalent characterization in terms of 3-valued logical consequence. We refer the reader to (Kunen, 1987) for more details.

We use $\mathcal{P} \models_K \alpha$ to denote that the Kunen semantics for the program \mathcal{P} supports α .

We can now relate the bottom-up characterization of a defeasible theory D with the Kunen semantics for the corresponding program \mathcal{D} .

Theorem 4 *Let D be a defeasible theory and \mathcal{D} denote its metaprogram counterpart.*

For each literal p ,

1. $D \vdash +\Delta p$ iff $\mathcal{D} \models_K \text{definitely}(p)$;
2. $D \vdash -\Delta p$ iff $\mathcal{D} \models_K \text{not_definitely}(p)$;
3. $D \vdash +\partial p$ iff $\mathcal{D} \models_K \text{defeasibly}(p)$;
4. $D \vdash -\partial p$ iff $\mathcal{D} \models_K \text{not_defeasibly}(p)$;

Thus Kunen’s semantics of \mathcal{D} characterizes the consequences of Defeasible Logic. Defeasible Logic can be decomposed into \mathcal{M} and Kunen’s semantics.

This has a further interesting implication: The consequences of predicate Defeasible Logic are computable. That is, if we permit predicates and uninterpreted function symbols of arbitrary arity, then the four sets of consequences are recursively enumerable. This follows from the fact that Kunen’s semantics is recursively enumerable (Kunen, 1989). It contrasts with most nonmonotonic logics, in which the consequence relation is not computable.

For propositional Defeasible Logic, we can use the relationship with Kunen’s semantics to establish a polynomial bound on the cost of computing consequences. In unpublished work we have a more precise bound.

Well-Founded Semantics

The presentation of the well-founded semantics in this section is based on (Van Gelder et al., 1991).

The notion of *unfounded sets* is the cornerstone of well-founded semantics. These sets provide the basis to derive negative conclusions in the well-founded semantics.

Definition 5 *Given a program \mathcal{P} , its Herbrand base H , and a partial interpretation I , a set $A \subseteq H$ is an unfounded set with respect to I iff each atom $\alpha \in A$ satisfies the following condition: For each instantiated rule R of \mathcal{P} whose head is α , (at least) one of the following holds:*

1. *Some subgoal of the body is false in I .*
2. *Some positive subgoal of the body occurs in A*

The *greatest unfounded set* of \mathcal{P} with respect to I ($U_{\mathcal{P}}(I)$) is the union of all the unfounded sets with respect to I .

Definition 6 *The transformations $T_{\mathcal{P}}(I)$, $U_{\mathcal{P}}(I)$, and $W_{\mathcal{P}}(I)$ are defined as follows:*

- $\alpha \in T_{\mathcal{P}}$ iff there is some instantiated rule R of \mathcal{P} such that α is the head of R , and each subgoal of R is true in I .
- $U_{\mathcal{P}}(I)$ is the greatest unfounded set with respect to I .
- $W_{\mathcal{P}} = T_{\mathcal{P}} \cup \neg U_{\mathcal{P}}(I)$, where $\neg U_{\mathcal{P}}(I)$ denotes the set obtained from $U_{\mathcal{P}}(I)$ by taking the complement of each atom in $U_{\mathcal{P}}(I)$.

We are now able to introduce the notion of *well-founded semantics*. The well-founded semantics of a program \mathcal{P} is represented by the least fixpoint of $W_{\mathcal{P}}$.

We write $\mathcal{P} \models_{WF} \alpha$ to mean that α receives the value t in the well-founded semantics of \mathcal{P} .

The following theorem establishes a correspondence between Well-Founded Defeasible Logic and the well-founded semantics of \mathcal{D} .

Theorem 7 *Let D be a defeasible theory and \mathcal{D} denote its metaprogram counterpart.*

For each ground literal p

1. $D \vdash_{WF} +\Delta p$ iff $\mathcal{D} \models_{WF} \text{definitely}(p)$;
2. $D \vdash_{WF} -\Delta p$ iff $\mathcal{D} \models_{WF} \text{not_definitely}(p)$;
3. $D \vdash_{WF} +\partial p$ iff $\mathcal{D} \models_{WF} \text{defeasibly}(p)$;
4. $D \vdash_{WF} -\partial p$ iff $\mathcal{D} \models_{WF} \text{not_defeasibly}(p)$;

Thus Well-Founded Defeasible Logic can be decomposed into \mathcal{M} and the well-founded semantics.

As a result, the consequences of a propositional Well-Founded Defeasible Logic theory can be computed in time polynomial in the size of the theory, using the fact that the well-founded semantics of propositional logic programs can be computed in polynomial time (Van Gelder et al., 1991). However, predicate Well-Founded Defeasible Logic is not computable, in contrast with predicate Defeasible Logic, which is computable.

Through the relationship between Kunen's semantics and the well-founded semantics of logic programs we can establish the relationship between Defeasible Logic and Well-Founded Defeasible Logic. The latter is an extension of Defeasible Logic in the sense that it respects the conclusions that are drawn by Defeasible Logic but generally draws more conclusions from the same syntactic theory.

Theorem 8 *Let D be a defeasible theory. For every conclusion C ,*

if $D \vdash C$ then $D \vdash_{WF} C$

Defeasible Logic with Explicit Failure

Although the meaning of conclusions $-\partial p$ and $-\Delta p$ is expressed in terms of failure-to-prove, there is not a way to express directly within the above defeasible logics that a literal should fail to be proved; tagged literals are not permitted in rules. In contrast, most logic programming-based formalisms employ "negation as failure" to express directly that a literal should fail.

The characterization of defeasible logics by a metaprogram and a semantics for logic programs provides a way for these logics to be extended with explicit failure without modifying their underlying semantics (i.e. a conservative extension). We introduce two operators on literals: $fail\Delta q$ which expresses that it should be proved that the literal q cannot be proven definitely, and $fail\partial q$ which expresses that it should be proved that q cannot be proven defeasibly. (Some syntactic restrictions must apply: classical negation (\neg) and the operators must not be applied to an operator expression, and $fail\partial$ is not permitted in strict rules.)

The meaning of such expressions can be given by appropriately modifying clauses $c2$, $c5$, $c6$ and $c7$ of the metaprogram. If an element Y_i of the body of a rule has the form $fail\partial Z_i$ then the body of $c5$ (say) should contain not $defeasibly(Z_i)$ in place of $defeasibly(Y_i)$. The resulting metaprogram is more general in that it defines a more expressive language but it has no different effect on theories that do not use the failure operators. Defeasible logic rules that do not involve explicit failure retain the same interpretation that they had before.

The resulting language, when we use the Kunen semantics, generalizes both Defeasible Logic and Courteous Logic Programs (Grosz, 1997). Indeed, it was already shown in (Antonioni et al., 1998) that Courteous Logic Programs can be expressed by Defeasible Logic theories, by encoding uses of the $fail$ operator with defeasible rules. With the explicit failure operators we can express $fail$ directly. Thus Courteous Logic Programs are essentially a syntactic subset of (with the same semantics as) the extended Defeasible Logic.

Future Work

This work opens up several variations of Defeasible Logic, in addition to the ones we have presented. The many different semantics for negation in logic programs have corresponding different semantics for the defeasible logic syntax. For example, the composition of stable model semantics (Gelfond and Lifschitz, 1988) with the metaprogram might produce a credulous version of defeasible logic.

Equally, we can vary the fundamental defeasible structure by modifying the metaprogram. Such changes will generally not alter the computational complexity, since it is the semantics of the meta-language which has the dominant effect on complexity.

The results also open up alternative implementations of defeasible logics. One possibility is to execute the metaprogram and data in a logic programming system with the appropriate semantics. The connection between Defeasible Logic and Kunen's semantics suggests an implementation incorporating constructive negation (Stuckey, 1995).

Conclusion

We have provided a semantic decomposition of defeasible logics into two parts: a metaprogram which specifies the fundamental defeasible structure of the logic (e.g. when a rule can be defeated by another), and a semantics which determines how the meta-language is interpreted.

We showed that Nute's Defeasible Logic is characterized by Kunen's semantics and that Well-Founded Defeasible Logic is characterized by the well-founded semantics. We also briefly developed a variant of Defeasible Logic with explicit failure by modifying the metaprogram. Thus different defeasible logics can be obtained by varying either the metaprogram or the semantics of the meta-language.

Decomposition is a useful tool for the analysis and comparison of logics. It provided a straightforward way to compare Defeasible Logic and Well-Founded Defeasible Logic. Equally, the reverse process of composition can be useful to design a logic with specific characteristics.

Acknowledgements

We thank Grigoris Antoniou and David Billington for discussions and comments on defeasible logic. This research was supported by the Australia Research Council under Large Grant No. A49803544.

References

- G. Antoniou, D. Billington and M.J. Maher. Normal Forms for Defeasible Logic. In *Proc. Joint International Conference and Symposium on Logic Programming*, J. Jaffar (Ed.), 160–174. MIT Press, 1998.
- D. Billington, K. de Coster and D. Nute. A Modular Translation from Defeasible Nets to Defeasible Logic. *Journal of Experimental and Theoretical Artificial Intelligence* 2 (1990): 151–177.
- D. Billington. Defeasible Logic is Stable. *Journal of Logic and Computation* 3 (1993): 370–400.
- M.A. Covington, D. Nute and A. Vellino. *Prolog Programming in Depth*. Prentice Hall 1997.
- M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Proc. Joint International Conference and Symposium on Logic Programming*, 1070–1080, MIT Press, 1988.
- B.N. Grosz. Prioritized Conflict Handling for Logic Programs. In *Proc. Int. Logic Programming Symposium*, J. Maluszynski (Ed.), 197–211. MIT Press, 1997.
- J.F. Horty, R.H. Thomason and D. Touretzky. A Skeptical Theory of Inheritance in Nonmonotonic Semantic Networks. In *Proc. AAAI-87*, 358–363.
- K. Kunen. Negation in Logic Programming. *Journal of Logic Programming* 4 (1987): 289–308.
- M. Maher, G. Antoniou and D. Billington. A Study of Provability in Defeasible Logic. In *Proc. Australian Joint Conference on Artificial Intelligence*, 215–226, LNAI 1502, Springer, 1998.
- D. Nute. Defeasible Reasoning. In *Proc. 20th Hawaii International Conference on Systems Science*, IEEE Press 1987, 470–477.
- D. Nute. Defeasible Logic. In D.M. Gabbay, C.J. Hogger and J.A. Robinson (eds.): *Handbook of Logic in Artificial Intelligence and Logic Programming Vol. 3*, Oxford University Press 1994, 353–395.
- P.J. Stuckey. Negation and Constraint Logic Programming. *Information and Computation* 118 (1995): 12–33.
- A. Van Gelder, K. Ross and J.S. Schlipf. Unfounded Sets and Well-Founded Semantics for General Logic Programs. *Journal of the ACM* 38 (1991): 620–650.