Embedding Defeasible Logic into Logic Programming

Grigoris Antoniou

Institute of Computer Science, FORTH, Greece
Dept. of Computer Science, University of Crete, Greece
Email: antoniou@ics.forth.gr

Abstract. Defeasible reasoning is a simple but efficient approach to nonmonotonic reasoning that has recently attracted considerable interest and that has found various applications. Defeasible logic and its variants are an important family of defeasible reasoning methods. So far no relationship has been established between defeasible logic and mainstream nonmonotonic reasoning approaches. In this paper we establish close links to known semantics of logic programs. In particular, we give a translation of a defeasible theory D into a meta-program $P(D)$. We show that under a condition of decisiveness, the defeasible consequences of D correspond exactly to the sceptical conclusions of $P(D)$ under the stable model semantics. Without decisiveness, the result holds only in one direction (all defeasible consequences of D are included in all stable models of $P(D)$). If we wish a complete embedding for the general case, we need to use the Kunen semantics of $P(D)$, instead.

1 Introduction

Defeasible reasoning is a nonmonotonic reasoning [20] approach in which the gaps due to incomplete information are closed through the use of defeasible rules that are usually appropriate. Defeasible logics were introduced and developed by Nute over several years [22]. These logics perform defeasible reasoning, where a conclusion supported by a rule might be overturned by the effect of another rule. Roughly, a proposition p can be defeasibly proved ($+\partial p$) only when a rule supports it, and it has been demonstrated that no applicable rule supports $\neg p$; this demonstration makes use of statements $-\partial q$ which mean intuitively that an attempt to prove q defeasibly has failed finitely. These logics also have a monotonic reasoning component, and a priority on rules. One advantage of Nute's design was that it was aimed at supporting efficient reasoning, and in our work we follow that philosophy.

Defeasible reasoning has recently attracted considerable interest. Its use in various application domains has been advocated, including the modelling of regulations and business rules [21, 13, 2], modelling of contracts [24], legal reasoning [23] and agent negotiations [11]. In fact, defeasible reasoning (in the form of courteous logic programs [12, 13]) provides a foundation for IBM's Business Rules Markup Language and for current W3C activities on rules. Therefore defeasible reasoning is arguably the most promising subarea in nonmonotonic reasoning as far as applications and integration to mainstram IT is concerned.

Recent theoretical work on defeasible logics has: (i) established some relationships to other logic programming approaches without negation as failure [3]; (ii) analysed

the formal properties of these logics [5, 18, 19], and (iii) has delivered efficient implementations [17].

However the problem remains that defeasible logic is not firmly linked to the mainstream of nonmonotonic reasoning, in particular the semantics of logic programs. This paper aims at resolving this problem. We use the translation of a defeasible theory D into a logic meta-program $\mathcal{M}$ proposed in [16]. For this translation we can show that

$$p \text{ is defeasibly provable in } D \iff p \text{ is included in all stable models of } \mathcal{M}. \quad (*)$$

However this result can only be shown under the additional condition of *decisiveness*: the absence of cycles in the atom dependency graph.

If we wish to drop decisiveness, $(*)$ holds only in one direction, from left to right. We show that if we wish the equivalence in the general case, we need to use another semantics for logic programs, namely Kunen semantics [15].

The paper is organised as follows. Sections 2 and 3 present the basics of defeasible logic and logic programming semantics, respectively. Section 4 presents our translation, while section 5 contains the main results. The proofs are found in an Appendix at the end of this paper.

2 Defeasible Logic

2.1 A Language for Defeasible Reasoning

A defeasible theory (a knowledge base in defeasible logic) consists of five different kinds of knowledge: facts, strict rules, defeasible rules, defeaters, and a superiority relation.

Facts are literals that are treated as known knowledge (given or observed facts of a case).

Strict rules are rules 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:

$$emu(X) \rightarrow bird(X).$$

Defeasible rules are rules that can be defeated by contrary evidence. An example of such a rule is "Birds typically fly"; written formally:

$$bird(X) \Rightarrow flies(X).$$

The idea is that if we know that something is a bird, then we may conclude that it flies, *unless there is other, not inferior, evidence suggesting that it may not fly.*

Defeaters are rules that cannot be used to draw any conclusions. Their only use is to prevent some conclusions. In other words, they are used to defeat some defeasible rules by producing evidence to the contrary. An example is "If an animal is heavy then it might not be able to fly". Formally:

$$heavy(X) \rightsquigarrow \neg flies(X).$$

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 against the conclusion that a heavy animal flies. In other words, we don't wish to conclude $\neg flies$ if $heavy$, we simply want to prevent a conclusion $flies$.

The *superiority relation* among rules is used to define priorities among rules, that is, where one rule may override the conclusion of another rule. For example, given the defeasible rules

$$r: \quad bird(X) \Rightarrow flies(X)$$
$$r': brokenWing(X) \Rightarrow \neg flies(X)$$

which contradict one another, no conclusive decision can be made about whether a bird with broken wings can fly. But if we introduce a superiority relation $>$ with $r' > r$, with the intended meaning that r' is strictly stronger than r, then we can indeed conclude that the bird cannot fly.

It is worth noting that, in defeasible logic, priorities are *local* in the following sense: Two rules are considered to be competing with one another only if they have complementary heads. Thus, since the superiority relation is used to resolve conflicts among competing rules, it is only used to compare rules with complementary heads; the information $r > r'$ for rules r, r' without complementary heads may be part of the superiority relation, but has no effect on the proof theory.

2.2 Formal Definition

In this paper we restrict attention to essentially propositional defeasible logic. Rules with free variables are interpreted as rule schemas, that is, as the set of all ground instances; in such cases we assume that the Herbrand universe is finite. We assume that the reader is familiar with the notation and basic notions of propositional logic. If q is a literal, $\sim q$ denotes the complementary literal (if q is a positive literal p then $\sim q$ is $\neg p$; and if q is $\neg p$, then $\sim q$ is p).

Rules are defined over a *language* (or *signature*) Σ, the set of propositions (atoms) and labels that may be used in the rule.

A *rule* $r : A(r) \hookrightarrow C(r)$ consists of its unique *label* r, its *antecedent* $A(r)$ ($A(r)$ may be omitted if it is the empty set) which is a finite set of literals, an arrow $\hookrightarrow$ (which is a placeholder for concrete arrows to be introduced in a moment), and its *head* (or *consequent*) $C(r)$ which is a literal. In writing rules we omit set notation for antecedents and sometimes we omit the label when it is not relevant for the context. There are three kinds of rules, each represented by a different arrow. Strict rules use $\rightarrow$, defeasible rules use $\Rightarrow$, and defeaters use $\rightsquigarrow$.

Given a set R of rules, we denote the set of all strict rules in R by R_s, and the set of strict and defeasible rules in R by R_{sd}. $R[q]$ denotes the set of rules in R with consequent q.

A *superiority relation on* R is a relation $>$ on R. When $r_1 > r_2$, then r_1 is called *superior* to r_2, and r_2 *inferior* to r_1. Intuitively, $r_1 > r_2$ expresses that r_1 overrules r_2, should both rules be applicable. $>$ must be acyclic (that is, its transitive closure must be irreflexive).

A *defeasible theory* D is a triple $(F, R, >)$ where F is a finite set of facts, R a finite set of rules, and $>$ an acyclic superiority relation on R.

A defeasible theory D is called *decisive* iff the atom dependency graph of D is acyclic.

Proposition 1. *[7] If D is decisive, then for each literal p:*

(a) either $D \vdash +\Delta p$ or $D \vdash -\Delta p$
(b) either $D \vdash -\partial p$ or $D \vdash +\partial p$.

Not every defeasible theory satisfies this property. For example, in the theory consisting of the single rule

$$p \Rightarrow p$$

neither $-\partial p$ nor $+\partial p$ is provable.

2.3 Proof Theory

A *conclusion* of a defeasible theory D is a tagged literal. A conclusion has one of the following four forms:

- $+\Delta q$, which is intended to mean that the literal q is definitely provable, using only strict rules.
- $-\Delta q$, which is intended to mean that q is provably not definitely provable (finite failure).
- $+\partial q$, 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.

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 $P = P(1), \ldots, P(n)$ of tagged literals satisfying the following conditions. The conditions are essentially inference rules phrased as conditions on proofs. $P(1..i)$ denotes the initial part of the sequence P of length i.

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

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

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

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.

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

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

Defeasible logic is closely related to several non-monotonic logics [2]. In particular, the "directly skeptical" semantics of non-monotonic inheritance networks [14] can be considered an instance of inference in DL once an appropriate superiority relation, derived from the topology of the network, is fixed [6].

3 Semantics of Logic Programs

A *logic program* P is a finite set of program clauses. A *program clause* r has the form

$$A \leftarrow B_1, \ldots, B_n, not\ C_1, \ldots, not\ C_m$$

where $A, B_1, \ldots B_n, C_1, \ldots, C_m$ are positive literals. A program clause with variables is considered to represent the set of its ground instances.

3.1 Stable Model Semantics

Let M be a subset of the Herbrand base. We call a ground program clause

$$A \leftarrow B_1, \ldots, B_n, not\ C_1, \ldots, not\ C_m$$

irrelevant w.r.t. M if at least one C_i is included in M. Given a logic program P, we define the *reduct of* P *w.r.t.* M, denoted by P^M, to be the logic program obtained from $ground(P)$ by

1. removing all clauses that are irrelevant w.r.t. M, and
2. removing all premises *not* C_i from all remaining program clauses.

Note that the reduct P^M is a definite logic program, and we are no longer faced with the problem of assigning semantics to negation, but can use the least Herbrand model instead. M is a *stable model* of P iff $M = \mathcal{M}_{P^M}$.

3.2 Kunen Semantics

Kunen semantics [15] is a 3-valued semantics for logic programs. A *partial interpretation* is a mapping from ground atoms to one of the three truth values **t**, **f** and **u**, which denote true, false and unknown, respectively. This mapping can be extended to arbitrary formulas using Kleene's 3-valued logic.

Kleene's truth tables can be summarized as follows. If φ is a boolean combination of atoms with truth value one of **t**, **f** and **u**, its truth value is **t** iff all possible ways of putting **t** or **f** for the various **u**-values lead to a value **t** being computed in ordinary (2-valued) logic; φ gets the value **f** iff $\neg\varphi$ gets the value **t**; 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 conjunctions or disjunctions.

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

1. $I_0(a) = \mathbf{u}$ for every atom a.
2. $I_{n+1}(a) = \mathbf{t}$ iff there is a program clause
   $A \leftarrow B_1, \ldots, B_n, not\ C_1, \ldots, not\ C_m$
   and a ground substitution σ such that $a = A\sigma$ and that
   $I_n((B_1 \wedge \ldots \wedge B_n \wedge \neg C_1 \wedge \ldots \wedge \neg C_m)\sigma) = \mathbf{t}$.
3. $I_{n+1}(a) = \mathbf{f}$ iff for all clauses
   $A \leftarrow B_1, \ldots, B_n, not\ C_1, \ldots, not\ C_m$
   in the program, and all ground substitutions σ, if $a = A\sigma$ then
   $I_n((B_1 \wedge \ldots \wedge B_n \wedge \neg C_1 \wedge \ldots \wedge \neg C_m)\sigma) = \mathbf{f}$.
4. $I_{n+1}(a) = \mathbf{u}$ if neither 2. nor 3. applies.

We shall say that the Kunen semantics of P supports a formula φ, written $P \models_K \varphi$, iff there is an interpretation I_n, for some finite n, such that $I_n(\varphi) = \mathbf{t}$.

4 A Translation of Defeasible Theories into Logic Programs

In this section we describe a meta-program $\mathcal{M}$ in a logic programming form that expresses the essence of the defeasible reasoning embedded in defeasible logic first introduced in [16]. $\mathcal{M}$ 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).
```

Next we introduce 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 the proof theory.

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

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

c3   defeasibly(X):-
       definitely(X).

c4   defeasibly(X):-
       not definitely(~ X),
       supportive_rule(R, X, [Y1, ..., Yn]),
       defeasibly(Y1),...,defeasibly(Yn),
       not overruled(R, X).

c5   overruled(R, X):-
       rule(S, ~ X, [U1, ..., Un]),
       defeasibly(U1),...,defeasibly(Un),
       not defeated(S, ~ X).

c6   defeated(S, ~ X):-
       sup(T, S),
       supportive_rule(T, X, [V1, ..., Vn]),
       defeasibly(V1),...,defeasibly(Vn).
```

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

We have permitted ourselves some syntactic flexibility in presenting the meta-program. However, there is no technical difficulty in using conventional logic programming syntax to represent this program.

Finally, for a defeasible theory $D = (F, R, >)$, we add facts according to the following guidelines:

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

5 Properties of the Translation

We establish relationships between D and its translation $\mathcal{M}$. To do so we must select appropriate logic program semantics to interpret *not*. First we consider stable model semantics.

Theorem 1.

(a) $D \vdash +\Delta p \Rightarrow definitely(p)$ *is included in all stable models of* $\mathcal{M}$.
(b) $D \vdash -\Delta p \Rightarrow definitely(p)$ *is not included in any stable model of* $\mathcal{M}$.
(c) If D *is decisive then the implications (a) and (b) are also true in the opposite direction.*

Theorem 2.

(a) $D \vdash +\partial p \Rightarrow defeasibly(p)$ *is included in all stable models of* $\mathcal{M}$.
(b) $D \vdash -\partial p \Rightarrow defeasibly(p)$ *is not included in any stable model of* $\mathcal{M}$.
(c) If D *is decisive then the implications (a) and (b) are also true in the opposite direction.*

That is, if D is decisive, then the stable model semantics of $\mathcal{M}$ corresponds to the provability in defeasible logic. However part (c) is not true in the general case, as the following example shows.

Example 1. Consider the defeasible theory

$r_1 : \Rightarrow \neg p$
$r_2 : \ p \Rightarrow p$

In defeasible logic, $+\partial\neg p$ cannot be proven because we cannot derive $-\partial p$. However, $defeasibly(\neg p)$ is a sceptical conclusion of $\mathcal{M}$ under stable model semantics because it is included in the only stable model of $\mathcal{M}$.

If we wish to have an equivalence result without the condition of decisiveness, then we must use a different logic programming semantics, namely Kunen semantics.

Theorem 3. *[16]*

(a) $D \vdash +\Delta p \Leftrightarrow \mathcal{M} \models_K definitely(p)$.
(b) $D \vdash -\Delta p \Leftrightarrow \mathcal{M} \models_K \neg definitely(p)$.
(c) $D \vdash +\partial p \Leftrightarrow \mathcal{M} \models_K defeasibly(p)$.
(b) $D \vdash -\partial p \Leftrightarrow \mathcal{M} \models_K \neg defeasibly(p)$.

6 Conclusion

We motivated and presented a translation of defeasible theories into logic programs, such that the defeasible conclusions of the former correspond exactly with the sceptical conclusions of the latter under the stable model semantics, if a condition of decisiveness is satisfied. If decisiveness is not satisfied, we have to use Kunen semantics instead.

This paper closes an important gap in the theory of nonmonotonic reasoning, in that it relates defeasible logic with mainstream semantics of logic programming. This result is particularly important, since defeasible reasoning is one of the most successful nonmonotonic reasoning paradigms in applications.

Acknoweldgements

I am grateful to David Billington, Guido Governatori and Michael Maher for fruitful discussions on defeasible logic and its relationship to logic programming.

References

1. G. Antoniou. *Nonmonotonic Reasoning*. MIT Press, (1997)
2. G. Antoniou, D. Billington and M.J. Maher. On the Analysis of Regulations Using Defeasible Rules. *Proceedings 32nd Hawaii International Conference on Systems Science*, (1999)
3. G. Antoniou, M.J. Maher and D. Billington. Defeasible Logic versus Logic Programming without Negation as Failure, *Journal of Logic Programming* 42:47-57 (2000)
4. G. Antoniou, D. Billington, G. Governatori and M.J. Maher. A Flexible Framework for Defeasible Logics. *Proceedings 17th American National Conference on Artificial Intelligence (AAAI2000)*, (2000) 405-410
5. G. Antoniou, D. Billington, G. Governatori and M.J. Maher. Representation Results for Defeasible Logic. *ACM Transactions on Computational Logic*, 2:255-287 (2001)
6. D. Billington, K. de Coster and D. Nute. A Modular Translation from Defeasible Nets to Defeasible Logic. *Journal of Experimental and Theoretical Artificial Intelligence* 2:151-177 (1990)
7. D. Billington. Defeasible Logic is Stable. *Journal of Logic and Computation* 3:370-400 (1993)
8. J.P. Delgrande, T. Schaub and H. Tompits. Logic Programs with Compiled Preferences. *Proceedings ECAI Conference* (2000) 464-468
9. M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. *Proceedings International Conference on Logic Programming*, MIT Press (1988) 1070-1080
10. M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Deductive Databases. *New Generation Computing* 9:365-385 (1991)
11. G. Governatori, A. ter Hofstede and P. Oaks. Defeasible Logic for Automated Negotiation. *Proceedings 5th CollECTeR Conference on Electronic Commerce*, Brisbane (2000)
12. B.N. Grosof. Prioritized Conflict Handling for Logic Programs. *Proceedings International Logic Programming Symposium*, MIT Press (1997) 197-211
13. B.N. Grosof, Y. Labrou and H.Y. Chan. A Declarative Approach to Business Rules in Contracts: Courteous Logic Programs in XML. *Proceedings 1st ACM Conference on Electronic Commerce (EC99)*, ACM Press (1999)
14. J.F. Horty, R.H. Thomason and D. Touretzky. A Skeptical Theory of Inheritance in Nonmonotonic Semantic Networks. *Proceedings AAAI Conference*, (1987) 358-363

15. K. Kunen. Negation in Logic Programming. *Journal of Logic Programming* 4:289-308 (1987)
16. M. Maher and G. Governatori. A Semantic Decomposition of Defeasible Logics. *Proceedings American National Conference on Artificial Intelligence (AAAI)*, AAAI/MIT Press (1999) 299-305
17. M.J. Maher, A. Rock, G. Antoniou, D. Billington and T. Miller. Efficient Defeasible Reasoning Systems. *Proceedings 12th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, (2000) 384-392
18. M.J. Maher. A Denotational Semantics for Defeasible Logic. *Proceedings 1st International Conference on Computational Logic*, Springer LNAI Vol.1861, (2000) 209-222
19. M.J. Maher. Propositional Defeasible Logic has Linear Complexity. *Theory and Practice of Logic Programming*, to appear.
20. V. Marek and M. Truszczynski. *Nonmonotonic Logic*. Springer (1993)
21. L. Morgenstern. Inheritance Comes of Age: Applying Nonmonotonic Techniques to Problems in Industry. *Artificial Intelligence*, 103:1-34 (1998)
22. 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
23. H. Prakken. *Logical Tools for Modelling Legal Argument: a Study of Defeasible Reasoning in Law*. Kluwer Academic Publishers (1997)
24. D.M. Reeves, B.N. Grosof, M.P. Wellman, and H.Y. Chan. Towards a Declarative Language for Negotiating Executable Contracts, *Proceedings AAAI99 Workshop on Artificial Intelligence in Electronic Commerce (AIEC-99)*, AAAI Press / MIT Press, (1999)
25. R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence* 13:81-132, (1980)