TLP **1** (6): 691–711, November 2001. © 2001 Cambridge University Press
DOI: 10.1017/S1471068401001168 Printed in the United Kingdom

Propositional defeasible logic has linear complexity

MICHAEL J. MAHER
Department of Mathematical and Computer Sciences, Loyola University Chicago, Chicago, IL, USA
(*e-mail:* `mjm@cs.luc.edu`)

School of Computing & Information Technology, Griffith University

Abstract

Defeasible logic is a rule-based nonmonotonic logic, with both strict and defeasible rules, and a priority relation on rules. We show that inference in the propositional form of the logic can be performed in linear time. This contrasts markedly with most other propositional nonmonotonic logics, in which inference is intractable.

KEYWORDS: defeasible reasoning, non-monotonic logic, computational complexity

1 Introduction

Most work in non-monotonic reasoning has focussed on languages for which propositional inference is not tractable. Sceptical default reasoning is Π_2^p-hard, even for very simple classes of default rules, as is sceptical autoepistemic reasoning and propositional circumscription. The complexity of sceptical inference from logic programs with negation-as-failure varies according to the semantics of negation. For both the stable model semantics and the Clark completion, sceptical inference is co-NP-hard. See Gottlob (1992) and Cadoli & Schaerf (1993) for more details.

Although such languages are very expressive, and this expressiveness has been exploited in answer-set programming (Niemelä, 1999), for example, they have not led to any practical applications in non-monotonic reasoning.

There has also been work on more tractable languages. Extensive work on inheritance networks with exceptions has led to polynomial time algorithms (Stein, 1992) and applications (Morgenstern, 1998). Defeasible logic (Nute, 1987; Nute, 1994) is a generalization (Billington *et al.*, 1990) of inheritance networks with exceptions under the directly sceptical semantics (Horty, 1994). Defeasible logic replaces the implicit specificity relation by an explicit, programmable priority relation; generalizes containment statements among concepts and their complements to rules over literals; and adds the notion of an explicit defeater. Recent work has proposed other languages (Grosof, 1997; Dimopoulos & Kakas, 1995) which are essentially subsets of defeasible logic (Antoniou *et al.*, 2000).

We have already established that full first-order defeasible logic has a recursively enumerable inference problem (Maher & Governatori, 1999). In this paper we establish that inference in propositional defeasible logic has linear complexity. Considering the several expressive features of defeasible logic, and its low complexity, it appears to be a suitable basis for applications of non-monotonic reasoning that involve very large rule sets.

In the next section we introduce the proof theory of defeasible logic, which defines the logic. Then we present a transition system (for a subset of defeasible logic) that progressively simplifies a defeasible theory while accumulating conclusions that can be inferred from the theory. We show that the meaning of the theory is preserved, and when no more transitions are applicable all consequences have been accumulated.

The following section presents an algorithm that can be viewed as performing a particular sequence of transitions. By an appropriate choice of data structures, we show that the set of all conclusions can be computed in time linear in the size of the theory. Finally, we use the transformations of Antoniou *et al.* (1998, 2001) to map an arbitrary defeasible theory to the subset of defeasible logic to which the algorithm applies. These transformations require only linear time, and blowup in the size of the theory is linear. Thus the result is a linear time computation of the conclusions of the defeasible theory. This algorithm has been the basis of an implementation of defeasible logic (Maher *et al.*, 2000).

2 Defeasible logic

2.1 An informal presentation

We begin by presenting the basic ingredients of defeasible logic. A defeasible theory (a knowledge base in defeasible logic, or a defeasible logic program) consists of five different kinds of knowledge: facts, strict rules, defeasible rules, defeaters, and a superiority relation.

Facts are indisputable statements, for example, “Tweety is an emu”. Written formally, this would be expressed as $emu(tweety)$.

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 does not fly. It is only evidence that the animal *may* not be able to fly. In other words, we don’t wish to conclude $\neg flies(X)$ if $heavy(X)$, we simply want to prevent a conclusion $flies(X)$ unless there is more evidence that overrides the heaviness of X.

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

$$\begin{array}{rrl} r: & bird(X) & \Rightarrow flies(X) \\ r': & brokenWing(X) & \Rightarrow \neg flies(X) \end{array}$$

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.

There are several relevant points to be made about the superiority relation in defeasible logic. Notice that a cycle in the superiority relation is counter-intuitive. In the above example, it makes no sense to have both $r > r'$ and $r' > r$. Consequently, defeasible logic requires that the superiority relation is acyclic.

A second point worth noting is that, in defeasible logic, priorities are *local* in the following sense: Two rules are considered to be competing with one another exactly when 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.

Finally, sets of rules with competing heads work as *teams* against each other. For example, consider the following rules about mammals

$$\begin{array}{rrl} r_1: & monotreme(X) & \Rightarrow mammal(X) \\ r_2: & hasFur(X) & \Rightarrow mammal(X) \\ r_3: & laysEggs(X) & \Rightarrow \neg mammal(X) \\ r_4: & hasBill(X) & \Rightarrow \neg mammal(X) \end{array}$$

with the following superiority relation: $r_1 > r_3$ and $r_2 > r_4$. For a platypus, which satisfies all antecedents, no rule is superior to all competing rules. However, rules r_1 and r_2 together are superior to rules r_3 and r_4. Hence defeasible logic concludes that a platypus is a mammal.

Defeasible logic has some similarities with default logic (Reiter, 1980): both are non-monotonic logics that distinguish statements which are unarguable (strict rules/ facts) from statements that have lesser force (defeasible rules/defaults). However, there are substantial syntactic and semantic differences.

First, default logic has no notion of priority among rules, nor of a defeater that cannot be used to make inferences. On the other hand, default logic permits any logical expression as a fact or in a default rule, whereas defeasible logic allows only rules as statements. Nevertheless, even if we restrict attention to propositional default theories (W, D) where W is a conjunction of literals and D consists of Horn default rules (so that the syntax is weaker than defeasible logic), sceptical inference is co-NP-hard (Cadoli & Schaerf, 1993) whereas, as will be shown in this paper, defeasible logic has linear complexity.

As noted by Nute (1994), a central difference between the two logics is in the application of rules. In default logic the applicability of a default rule is independent of any other default rule so that, given default rules $\frac{:a}{a}$ and $\frac{:\neg a}{\neg a}$, either default may be applied. On the other hand, in defeasible logic with rules $\{\Rightarrow a, \Rightarrow \neg a\}$ neither rule may be applied, since each rule 'interferes' with – or defeats – the other.

2.2 *Formal definition*

In this paper, we restrict attention to essentially propositional defeasible logic. Rules with free variables can be interpreted as rule schemas, that is, as the set of all variable-free instances; in such cases we assume that the Herbrand universe is finite and we regard variable-free atoms as propositions. For compactness of presentation, we use rule schemas in examples. However, in this paper we assume that a theory is presented in propositional form, and not with rule schemas.

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).

A *rule* $r : A(r) \hookrightarrow C(r)$ consists of its unique *label* r, its *antecedent* (or *body*) $A(r)$ ($A(r)$ 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 *consequent* (or *head*) $C(r)$ 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 $\leadsto$.

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, the set of defeasible rules and defeaters in R by R_{dd}, and the set of defeaters in R by R_{dft}. $R[q]$ 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. We assume $>$ to be acyclic (that is, the transitive closure of $>$ is irreflexive).

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

Example 1
We will use the following defeasible theory to demonstrate several aspects of defeasible logic. We assume there are only the constants *ethel* and *tweety* in the language.

Let $D_{bird} = (F_{bird}, R_{bird}, >_{bird})$ where: F_{bird} is the set of facts

$$\begin{array}{l} emu(ethel). \\ bird(tweety). \end{array}$$

R_{bird} is represented by the set of rule schemas

$$\begin{array}{lrcl} r_1: & emu(X) & \rightarrow & bird(X). \\ r_2: & bird(X) & \Rightarrow & flies(X). \\ r_3: & heavy(X) & \leadsto & \neg flies(X). \\ r_4: & brokenWing(X) & \Rightarrow & \neg flies(X). \\ r_5: & & \Rightarrow & heavy(ethel). \end{array}$$

and the superiority relation $>_{bird}$ contains only $r_4 >_{bird} r_2$.

The five rule schemas give rise to nine propositional rules by instantiating each variable to *ethel* and *tweety*, respectively. Those rules, which make up R_{bird} are

$$\begin{array}{lrcl} r_{1,e}: & emu(ethel) & \rightarrow & bird(ethel). \\ r_{1,t}: & emu(tweety) & \rightarrow & bird(tweety). \\ r_{2,e}: & bird(ethel) & \Rightarrow & flies(ethel). \\ r_{2,t}: & bird(tweety) & \Rightarrow & flies(tweety). \\ r_{3,e}: & heavy(ethel) & \leadsto & \neg flies(ethel). \\ r_{3,t}: & heavy(tweety) & \leadsto & \neg flies(tweety). \\ r_{4,e}: & brokenWing(ethel) & \Rightarrow & \neg flies(ethel). \\ r_{4,t}: & brokenWing(tweety) & \Rightarrow & \neg flies(tweety). \\ r_5: & & \Rightarrow & heavy(ethel). \end{array}$$

The rules have been re-labelled purely to simplify later reference[1]. As a result, the superiority relation becomes $\{r_{4,e} >_{bird} r_{2,e}, r_{4,e} >_{bird} r_{2,t}, r_{4,t} >_{bird} r_{2,e}, r_{4,t} >_{bird} r_{2,t}\}$. As noted in the previous subsection, the two statements $r_{4,e} >_{bird} r_{2,t}$ and $r_{4,t} >_{bird} r_{2,e}$ have no effect, since they do not involve rules with conflicting heads.

In this defeasible theory, $R_s = \{r_{1,e}, r_{1,t}\}$ and $R_d[\neg flies(tweety)] = \{r_{3,t}, r_{4,t}\}$. $\square$

2.3 *Proof theory*

Defeasible logic is defined by the proof theory presented in this subsection following (Billington, 1993). There are also characterizations of defeasible logic in other frameworks (Maher & Governatori, 1999; Governatori & Maher, 2000; Maher, 2000), but they are not needed here.

A *conclusion* of D is a tagged literal that may be proved by D, 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.

[1] Without the re-labelling we would have different rules with the same label. This is not a problem, formally, but might be confusing.

$+\partial q$ 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.

Conclusions are used only at the meta-level and do not occur in defeasible theories. Notice the distinction between $-$, which is used only to express unprovability, and $\neg$, which expresses classical negation. For example, $-\Delta\neg flies(tweety)$ means that it has been proved that the negated proposition $\neg flies(tweety)$ cannot be proved definitely in the defeasible theory.

If we are able to prove q definitely, then q is also defeasibly provable. This is a direct consequence of the formal definition below. Similarly, if $-\partial q$ is concluded then we must also conclude $-\Delta q$.

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 $P = (P(1), \ldots P(n))$ of tagged literals constructed by inference rules. There are four inference rules (corresponding to the four kinds of conclusion) that specify how a derivation may be extended. ($P(1..i)$ denotes the initial part of the sequence P of length i):

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

This 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 also be taken into account). From D_{bird} in Example 1 we can infer $+\Delta emu(ethel)$ (and $+\Delta bird(tweety)$) immediately, in a proof of length 1. Using $r_{1,e}$ and the second clause of the inference rule, we can infer $+\Delta bird(ethel)$ in a proof of length 2.

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

To prove $-\Delta q$, that is, 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 element a of the antecedent for which we have established that a is not definitely provable ($-\Delta a$). From D_{bird}, we can infer $-\Delta heavy(tweety)$ and $-\Delta\neg flies(tweety)$ immediately (among many others), since in these cases $R_s[q]$ is empty.

Note that this definition of nonprovability does not involve loop detection. Thus if D consists of the single rule $p \rightarrow p$, we can see that p cannot be proven, but defeasible logic is unable to prove $-\Delta p$.

Defeasible provability requires consideration of chains of reasoning for the complementary literal, and possible resolution using the superiority relation. Thus the inference rules for defeasible provability are more complicated than those for definite provability.

$+\partial$: We may append $P(i+1) = +\partial q$ if 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 "attacks", 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 in subsection 2.1). 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 s. Thus each attack on the conclusion q must be counterattacked by a stronger rule.

From D_{bird} in Example 1 we can infer $+\partial bird(ethel)$ in a proof of length 3, using part (1) of the inference rule. Other applications of this inference rule first require application of the inference rule $-\partial$. This inference rule completes the definition of the proof theory of defeasible logic. It is a strong negation of the inference rule $+\partial$ (Antoniou *et al.*, 2001).

$-\partial$: We may append $P(i+1) = -\partial q$ if
- (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 $t \not> s$

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 possibly applicable rule t with head q is superior to s (2.3).

From D_{bird} in Example 1 we can infer $-\partial brokenWing(ethel)$ with a proof of length

2 using (1) and (2.1), since $R[q]$ is empty. Employing this conclusion, we can then infer $-\partial\neg flies(ethel)$, again using (2.1), since the only rule r is $r_{4,e}$. Using r_5, we can infer $+\partial heavy(ethel)$ by (2) of the inference rule $+\partial$; (2.3) holds since $R[\sim q]$ is empty. Using (2.3) of the inference rule $-\partial$, we can infer $-\partial flies(ethel)$, where the role of s is taken by $r_{3,e}$.

Furthermore, we can infer $-\partial brokenWing(tweety)$ and $-\partial heavy(tweety)$ using (2.1) of inference rule $-\partial$, since in both cases $R_{sd}[q]$ is empty. Employing those conclusions, we can then infer $-\partial\neg flies(tweety)$, again using (2.1), since the only rules for $\neg flies(tweety)$ involve antecedents already established to be unprovable defeasibly. Now, using this conclusion, we can use (2), specifically (2.1), of the inference rule $+\partial$ to infer $+\partial flies(tweety)$.

The elements of a derivation P in D 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. Equivalently, we say that L is a *consequence* of D.

Conclusions are the basis of our notion of equivalent defeasible theories. We say D_1 and D_2 are *conclusion equivalent* (written $D_1 \equiv D_2$) iff D_1 and D_2 have identical sets of consequences, that is, $D_1 \vdash L$ iff $D_2 \vdash L$.

An important property of defeasible logic is that it is *coherent*, that is, there is no defeasible theory D and literal p such that $D \vdash +\partial p$ and $D \vdash -\partial p$, or $D \vdash +\Delta p$ and $D \vdash -\Delta p$ (Billington, 1993). Put simply, this property says that we cannot establish that a literal is simultaneously provable and unprovable.

Notice that strict rules are used in two different ways. When we try to establish *definite provability*, then strict rules are used as in classical logic: if their bodies are proved definitely then their head is proved definitely, regardless of any reasoning chains with the opposite conclusion. But strict rules can also be used to show *defeasible provability*, given that some other literals are known to be defeasibly provable. In this case, strict rules are used exactly like defeasible rules. For example, a strict rule may have its body proved defeasibly, yet it may not fire because there is a rule with the opposite conclusion that is not weaker. Furthermore, strict rules are not automatically superior to defeasible rules. In this paper, we will choose to duplicate strict rules as defeasible rules, and require definite reasoning to use the strict rules (as always), while defeasible reasoning may use only defeasible rules. Because of the above facts, this duplication and separation of rules does not modify the consequences. When there is a duplicate defeasible rule for every strict rule in a defeasible theory, we say that the theory has *duplicated strict rules*.

In this paper we will focus on the subset of defeasible logic that involves no superiority statements and no defeaters, which we call *basic defeasible logic*. For basic defeasible logic the inference rules are simplified (Antoniou *et al*., 2001). We simplify them further by introducing two auxiliary tags for literals ($+\sigma$ and $-\sigma$) and corresponding inference rules.

$+\sigma$: We may append $P(i+1) = +\sigma q$ if
$$\exists r \in R_{sd}[q]\ \forall a \in A(r) : +\partial a \in P(1..i)$$

$-\sigma$: We may append $P(i+1) = -\sigma q$ if
$$\forall r \in R_{sd}[q]\ \exists a \in A(r) : -\partial a \in P(1..i)$$

These tags $(+\sigma, -\sigma)$ represent the ability/inability to find a tentative reason for the literals. Essentially, we can prove $+\sigma q$ when there is an argument for q, but we must consider all counter-arguments (for $\sim q$) before we can prove $+\partial q$. Similarly, we can prove $-\sigma q$ if there is not even a tentative reason for q, but there are other ways to conclude $-\partial q$. Tagged literals involving σ, ∂ or Δ are called *extended conclusions.*

With the addition of these tags and inference rules, the inference rules for ∂ are reduced to:

$+\partial$: We may append $P(i+1) = +\partial q$ if either
$+\Delta q \in P(1..i)$ or
$\{+\sigma q, -\Delta{\sim}q, -\sigma{\sim}q\} \subseteq P(1..i)$

$-\partial$: We may append $P(i+1) = -\partial q$ if
$-\Delta q \in P(1..i)$ and
$\{-\sigma q, +\Delta{\sim}q, +\sigma{\sim}q\} \cap P(1..i) \neq \emptyset$

This modified proof system reduces the amount of 'work' done in any one proof step in comparison with the original inference rules, and is the logical basis for the algorithm. However, the algorithm is also based on repeated simplification of the defeasible theory, which we describe as a transition system in the next section.

3 The transition system

The algorithm is defined in terms of a transition system on states. A *state* is a pair (D, C) of a defeasible theory D and a set of extended conclusions C. As the algorithm proceeds, the theory D is simplified and consequences of the theory are accumulated in C. The transitions for the positive conclusions are based on forward chaining. The negative conclusions are derived by a dual process.

Definition 1
There is a transition $(D_i, C_i) \Longrightarrow (D_{i+1}, C_{i+1})$ only in the following cases:

1. $(D_i, C_i) \Longrightarrow (D_i, C_i \cup \{+\Delta q, +\partial q, +\sigma q\})$
   if there is a fact q in D_i or a strict rule in D_i with head q and empty body
2. $(D_i, C_i) \Longrightarrow (D_i, C_i \cup \{+\sigma q\})$
   if there is a defeasible rule in D_i with head q and empty body
3. $(D_i, C_i) \Longrightarrow (D_i, C_i \cup \{-\Delta q\})$
   if there is no strict rule in D_i with head q and no fact q in D_i
4. $(D_i, C_i) \Longrightarrow (D_i, C_i \cup \{-\sigma q\})$
   if there is no rule in D_i with head q
5. $(D_i, C_i) \Longrightarrow (D_i, C_i \cup \{+\partial q\})$
   if $+\Delta q \in C_i$ or if $\{-\Delta{\sim}q, -\sigma{\sim}q, +\sigma q\} \subseteq C_i$
6. $(D_i, C_i) \Longrightarrow (D_i, C_i \cup \{-\partial q\})$
   if $-\Delta q \in C_i$ and either $+\Delta{\sim}q \in C_i$ or $+\sigma{\sim}q \in C_i$ or $-\sigma q \in C_i$.
7. $(D_i, C_i) \Longrightarrow ((D_i \backslash \{r\}) \cup \{r'\}, C_i)$
   if r is a strict rule in D_i with body containing q, r' is r with q deleted, and $+\Delta q \in C_i$

8. $(D_i, C_i) \Longrightarrow ((D_i \backslash \{r\}) \cup \{r'\}, C_i)$
   if r is a defeasible rule in D_i with body containing q, r' is r with q deleted, and $+\partial q \in C_i$
9. $(D_i, C_i) \Longrightarrow ((D_i \backslash \{r\}), C_i)$
   if r is a strict rule in D_i with body containing q, and $-\Delta q \in C_i$
10. $(D_i, C_i) \Longrightarrow ((D_i \backslash \{r\}), C_i)$
   if r is a defeasible rule in D_i with body containing q, and $-\partial q \in C_i$

We write $(D_i, C_i) \not\Longrightarrow$ if there is no transition $(D_i, C_i) \Longrightarrow (D_{i+1}, C_{i+1})$ except those where $(D_i, C_i) = (D_{i+1}, C_{i+1})$.

If $(D, \emptyset) \Longrightarrow \cdots \Longrightarrow (D', C') \not\Longrightarrow$ and C is the subset of C' involving only the tags $+\Delta, -\Delta, +\partial$, and $-\partial$, then we say that D *derives* C.

We demonstrate the action of the transition system on D_{bird} in Example 1. Transition 1 applies to each fact and transition 3 applies to all literals except the facts and $bird(ethel)$. Similarly, transition 2 applies to $heavy(ethel)$ and transition 4 applies to several literals, including all $brokenWing$ literals. These transitions do not modify D_i; they only accumulate conclusions in C_i. However, as a result of this accumulation, transition 7 deletes $emu(ethel)$ from $r_{1,e}$, which then allows transition 1 to add $+\partial bird(ethel)$ to C_i, and transition 9 deletes $r_{1,t}$. Furthermore, transition 5 applies for $emu(ethel)$, $bird(tweety)$, and $bird(ethel)$ by the first condition, and then transition 8 applies to modify $r_{2,e}$ and $r_{2,t}$; similarly, transition 6 applies to all $brokenWing$ literals, which then allows $r_{4,e}$ and $r_{4,t}$ to be deleted by transition 10. Further transitions are possible.

It is quite clear that, by ignoring defeaters and superiority statements, the transition system is incorrect for theories of full defeasible logic. The transition system is also incomplete, in general, for theories of basic defeasible logic. For example, from the rules

$$\begin{array}{ll} a & \rightarrow b \\ & \Rightarrow a \end{array}$$

the transition system cannot conclude $+\sigma b$. But under the assumption that strict rules are duplicated as defeasible rules, the transition system is sound and complete, as we now show.

Theorem 1
Let D be a basic defeasible theory with duplicated strict rules. Suppose $(D, \emptyset) \Longrightarrow \cdots \Longrightarrow (D', C') \not\Longrightarrow$. If $c \in C'$ then $D \vdash c$.

Proof
Consider a transition $(D_i, C_i) \Longrightarrow (D_{i+1}, C_{i+1})$. We prove, by induction on i, that

- If $c \in C_{i+1} \backslash C_i$ then $D_i \vdash c$
- $D_i \equiv D_{i+1}$

for every conclusion c.

The first part is straightforward, since the first six transitions are essentially instances of the inference rules of defeasible logic. The remainder of the transitions do not modify C_i.

For the second part only the last four transitions are relevant. We need to show that, for each transition and every proof in D_i there is a corresponding proof in D_{i+1}, and vice versa.

For transition 7
If a proof p in D_i applies the inference rule $+\Delta$ using r, then the inference rule is still applicable using r' and p is also a proof in D_{i+1}. Similarly, if a proof p in D_i applies the inference rule $-\Delta$ involving r, then p establishes $-\Delta x$, for some x in the body of r. x cannot be q, since $D_i \vdash +\Delta q$. Thus p is also a proof in D_{i+1}.

If a proof p in D_{i+1} applies the inference rule $+\Delta$ using r', let p' be a proof in D_i of $+\Delta q$. Then the concatenation of p' and p is a proof in D_i. If a proof p in D_{i+1} applies the inference rule $-\Delta$ involving r', then p is also a proof in D_i.

For transition 8
Let s be the head of r. If p is a proof in D_i containing $+\sigma s$ then p is also a proof in D_{i+1}. If p is a proof in D_{i+1} containing $+\sigma s$, let p' be a proof in D_i of $+\partial q$. Then the concatenation of p' and p is a proof in D_i.

If p is a proof in D_i containing $-\sigma s$, then there is some literal t in the body of r such that $D_i \vdash -\partial t$. t cannot be q since, by assumption, $D_i \vdash +\partial q$, and defeasible logic is coherent. Thus p is also a proof in D_{i+1}. If p is a proof in D_{i+1} containing $-\sigma s$, then there is some literal t in the body of r' such that $D_{i+1} \vdash -\partial t$. We must also have $D_i \vdash -\partial t$, since the proof in D_{i+1} cannot use r', which would introduce circularity. Thus p is also a proof in D_i.

For transition 9
If a proof p in D_i applies the inference rule $+\Delta$ it does not use r since $D_i \vdash -\Delta q$. Thus p is also a proof in D_{i+1}. If a proof p in D_i applies the inference rule $-\Delta$ then the inference rule also applies in D_{i+1}. Thus p is also a proof in D_{i+1}.

If a proof p in D_{i+1} applies the inference rule $+\Delta$ then it is also a proof in D_i. If a proof p in D_{i+1} applies the inference rule $-\Delta$ let p' be a proof in D_i of $-\Delta q$. Then the concatenation of p' and p is a proof in D_i.

For transition 10
Let s be the head of r. If p is a proof in D_i containing $-\sigma s$ then p is also a proof in D_{i+1}. If p is a proof in D_{i+1} containing $-\sigma s$, let p' be a proof in D_i of $-\partial q$. Then the concatenation of p' and p is a proof in D_i.

If p is a proof in D_i containing $+\sigma s$, then the proof of $+\sigma s$ cannot use r, since $D_i \vdash -\partial q$. Thus p is also a proof in D_{i+1}. If p is a proof in D_{i+1} containing $+\sigma s$, then p is also a proof in D_{i+1}. $\square$

It follows from the above proof that the transformed defeasible theories are equivalent.

Corollary 1
Let D be a basic defeasible theory with duplicated strict rules. Suppose $(D, \emptyset) \Longrightarrow \cdots \Longrightarrow (D', C')$. Then $D \equiv D'$.

The transition system is complete: it generates all the conclusions that are inferred by defeasible logic.

Theorem 2

Let D be a basic defeasible theory with duplicated strict rules. Suppose $(D, \emptyset) \Longrightarrow \cdots \Longrightarrow (D', C') \not\Longrightarrow$. For every conclusion c, if $D' \vdash c$ then $c \in C'$.

Proof

Suppose, to obtain a contradiction, that the theorem does not hold, and let c be a consequence of D' that is not in C' and has a minimal length proof in D' among such consequences.

If c is $-\Delta q$ then every strict rule with head q has a body literal b such that $D' \vdash -\Delta b$. By the assumption, and since the proof for each $-\Delta b$ must be shorter, $-\Delta b \in C'$. Since no more transitions are possible, and considering transition 9, there is no strict rule with head q. But then, a transition 3 that adds $-\Delta q$ to C', is possible, contradicting the assumption.

If c is $+\Delta q$ then there is a strict rule with head q and $D' \vdash +\Delta b$ for each body literal b. By the assumption, and since the proof for $+\Delta b$ must be shorter, $+\Delta b \in C'$. Since no more transitions are possible, and considering transition 7, the body of the rule must be empty. But then, considering transition 1, $+\Delta q \in C'$, contradicting the assumption.

Similarly, if c is $+\partial q$ then there must be a defeasible rule for q with an empty body. Furthermore, $D' \vdash -\Delta \sim q$, and every defeasible rule for $\sim q$ contains a body literal b such that $D' \vdash -\partial b$. By the assumption, $-\partial b \in C'$, for each b, and $-\Delta \sim q \in C'$. Since no more transitions are possible, considering transition 10 there must be no defeasible rules for $\sim q$. Thus $-\sigma \sim q \in C'$. Also $+\sigma q \in C'$, since the rule body is empty, and thus the transition 5 applies, leading to a contradiction.

If c is $-\partial q$ then $D' \vdash -\Delta q$, and either every defeasible rule for q contains a body literal b such that $D' \vdash -\partial b$ or $D' \vdash +\Delta \sim q$ or and there is a defeasible rule for $\sim q$ such that for all its body literals b', $D' \vdash +\partial b'$. By the assumption, and since any of these proofs must be shorter than the proof of c, $-\Delta q \in C'$ and either every defeasible rule for q contains a body literal b such that $-\partial b \in C'$ or $+\Delta \sim q \in C'$ or and there is a defeasible rule for $\sim q$ such that for all its body literals b', $+\partial b' \in C'$. But then, since no transitions are possible, and considering transition 8 and 10, either there are no defeasible rules for q or $+\Delta \sim q \in C'$ or there is a defeasible rule for $\sim q$ with an empty body. Considering transitions 2 and 4, either $-\sigma q \in C'$ or $+\Delta \sim q \in C'$ or $+\sigma \sim q \in C'$. Consequently, because of transition 6, $-\partial q \in C'$, contradicting the original assumption. $\square$

Combining the previous two theorems, we can conclude that the transition system computes exactly the conclusions of the initial defeasible logic program.

Theorem 3

Let D be a basic defeasible theory with duplicated strict rules. Suppose D derives C. Then, for every conclusion c,

$D \vdash c$ iff $c \in C$

4 The linear algorithm

The algorithm consists of two main parts: (1) a series of transformations to produce an equivalent basic defeasible theory with duplicated strict rules; and (2) the application of transitions from the transition system in a computationally efficient order. These parts will be discussed in separate subsections.

4.1 Transformations

The input to the algorithm is an arbitrary defeasible theory, but the transition system is proved correct only for basic defeasible theories with duplicated strict rules. To bridge this gap we must transform the input defeasible theory into the appropriate form.

In Antoniou *et al.* (2001), three transformations are used to convert any defeasible theory into an equivalent basic defeasible theory. The first places the defeasible theory in a normal form, the second eliminates defeaters, and the third reduces the superiority relation to the empty relation. Let *Basic* denote this sequence of transformations. We will use *Basic* in the inference algorithm.

It is beyond the scope of this paper to present a definition of *Basic*; for that, the reader is referred to Antoniou *et al.* (2001). However, to give the flavor of the transformations we present one component transformation, *elim_sup*, which empties the superiority relation. It introduces two new propositions for each rule, two rules for each superiority statement, and up to four rules replacing each original rule. It is defined as follows.

Definition 2
Let D be a regular defeasible theory with rules R. Let Σ be the language of D.

$$\begin{aligned} \mathit{elim\_sup}(R) = R_s \quad & \cup \{ \quad \neg \mathit{inf}^+(r_1) \Rightarrow \mathit{inf}^+(r_2), \\ & \qquad \neg \mathit{inf}^-(r_1) \Rightarrow \mathit{inf}^-(r_2) \mid r_1 > r_2 \} \\ & \cup \{ \quad A(r) \Rightarrow \neg \mathit{inf}^+(r), \\ & \qquad \neg \mathit{inf}^+(r) \Rightarrow p, \\ & \qquad A(r) \Rightarrow \neg \mathit{inf}^-(r), \\ & \qquad \neg \mathit{inf}^-(r) \Rightarrow p \mid r \in R_d[p] \} \\ & \cup \{ \quad A(r) \Rightarrow \neg \mathit{inf}^-(r), \\ & \qquad \neg \mathit{inf}^-(r) \leadsto p \mid r \in R_{dft}[p] \} \end{aligned}$$

For each r, $\mathit{inf}^+(r)$ and $\mathit{inf}^-(r)$ are new atoms not in Σ. Furthermore, all new atoms are distinct.

Intuitively, the two introduced propositions, $\mathit{inf}^+(r)$ and $\mathit{inf}^-(r)$, represent that r is inferior to a rule in R_{sd} (respectively R) that has a defeasibly provable body. If $D \vdash +\sigma \mathit{inf}^+(r)$ then $D \not\vdash +\partial \neg \mathit{inf}^+(r)$ and even when $A(r)$ is proved defeasibly p, the consequent of r, cannot be established defeasibly using r because the link between the two is broken.

The transformation to eliminate defeaters is slightly similar to the above transformation; two propositions are introduced for every proposition in D and up to three rules are used to replace each rule in D. The two propositions – p^+ and

p^- – introduced for each proposition p, represent that p (respectively $\neg p$) is not defeated by a defeater. Again, these propositions form links between antecedent and consequent, and the effect of a defeater is obtained with a defeasible rule that can break such a link. For more details, see Antoniou *et al.* (2001).

It is clear that these two transformations are profligate in their introduction of propositions and generation of rules. Nevertheless, they can be applied in one pass (each) over a defeasible theory, and the resulting defeasible theory is, at most, a constant factor larger.

The following result was proved in Antoniou *et al.* (2001), albeit in parts and with slightly different terminology.

Theorem 4 (Antoniou *et al.*, 2001)

Let D be a defeasible theory and let Σ be the language of D. Let $D' = Basic(D)$. Then D' is a basic defeasible theory and, for all conclusions c in Σ,

$D \vdash c$ iff $D' \vdash c$.

Furthermore, D' can be constructed in time linear in the size of D, and the size of D' is linear in the size of D.

We must also ensure that the basic defeasible theory resulting from *Basic* has duplicated strict rules. To guarantee this property we employ the following simple transformation *DupStrict*[2].

Definition 3

Let $D = (F, R, >)$. Then $DupStrict(D) = (F, R', >)$ where

$$R' = R \cup \{(r : B \Rightarrow q) \mid (r : B \rightarrow q) \in R_s\}$$

It is straightforward to show that *DupStrict* does not change the consequences of a theory.

Proposition 1

For any defeasible theory D, and any conclusion c,

$D \vdash c$ iff $DupStrict(D) \vdash c$

Proof

(Sketch) Definite provability is affected only by strict rules and facts. The strict rules in R and R' are essentially the same, and the facts are unchanged. Thus definite provability is unchanged by *DupStrict*.

In applying the defeasible inference rules, it makes no difference whether a rule is strict or defeasible. So $\vdash$, applied to $DupStrict(D)$ uses effectively the same rules as when applied to D. It also makes no difference whether there is one or several copies of a rule. $\square$

```
$D' = (F', R', \emptyset) = Basic(D)$
$R = DupStrict(\ R'\ )$
initialize $S$
$K = \emptyset$

**while** ( $S \neq \emptyset$ )
    choose $s \in S$ and delete $s$ from $S$
    add $s$ to $K$
    **case** $s$ of
        $+\Delta p$:
            **delete** all occurrences of $p$ in all rule bodies
            **whenever** a body of a strict rule with head $h$ becomes empty
                add $+\Delta h$ to $S$
                record $+\Delta h, +\partial h, +\sigma h$
                $CheckInference(\ +\Delta h, S\ )$
            **whenever** a body of a defeasible rule with head $h$ becomes empty
                record $+\sigma h$
                $CheckInference(\ +\sigma h, S\ )$
        $-\Delta p$:
            **delete** all strict rules where $p$ occurs in the body
            **whenever** there are no more strict rules for a literal $h$, and there is no fact $h$
                add $-\Delta h$ to $S$
                record $-\Delta h$
                $CheckInference(\ -\Delta h, S\ )$
        $+\partial p$:
            **delete** all occurrences of $p$ in defeasible rule bodies
            **whenever** a body with head $h$ becomes empty
                record $+\sigma h$
                $CheckInference(\ +\sigma h, S\ )$
        $-\partial p$:
            **delete** all defeasible rules where $p$ occurs in the body
            **whenever** there are no more defeasible rules for a literal $h$
                record $-\sigma h$
                $CheckInference(\ -\sigma h, S\ )$
    **end case**
**end while**
```

Fig. 1. Inference algorithm for defeasible logic.

4.2 The algorithm

In the algorithm, which is presented in Figure 1, p ranges over literals and s ranges over conclusions. K and S are sets of conclusions. K accumulates the set of conclusions that have been proved and used, while S holds those proven conclusions that have not yet been used to establish more conclusions. D is the input defeasible theory.

[2] This duplication is not strictly necessary since an appropriate separation of definite and defeasible reasoning is achieved by the transformations of Antoniou *et al.* (2001). However, in this context, it is simpler to ensure this property by duplication than to argue that it holds of transformations that are not presented here.

To begin the algorithm we reduce D to basic defeasible logic with duplicated strict rules, as discussed above. We also initialize the set S, that is, we add to S those conclusions that can immediately be established: all facts are definitely true, as are the heads of strict rules with empty bodies. The heads of defeasible rules with empty bodies are tentatively true. Similarly, those literals with no (strict) rules for them are unprovable (strictly).

The algorithm proceeds by modifying the rules in the theory. For strict rules, when inferring positive definite consequences, the algorithm is similar to unit resolution for definite clauses in classical logic: when an atom is proved, it can be eliminated from the bodies of all other definite clauses. In this case, when a literal is established definitely it can be deleted from the body of all rules. Similarly, when it is established that a literal p cannot be proved then those strict rules which have p as a precondition cannot be used to prove the head, and so they can be deleted. When a strict rule has an empty body, then its head is definitely provable; when there are no strict rules for a literal, then the literal is unprovable definitely. When an atom is proved definitely it is eliminated from both strict and defeasible rules; this necessitates extra code for the case when the body of a defeasible rule becomes empty.

When establishing tentative provability, we proceed in exactly the same way as with definite provability, except that we restrict attention to defeasible rules. We can conclude $+\sigma q$ precisely when the body of a defeasible rule for q becomes empty, and $-\sigma q$ precisely when there are no more defeasible rules for q.

When establishing defeasible provability, we use the simplified inference rules introduced at the end of section 2. Each time a statement such as $+\sigma p$ is inferred by the system the statement is *record*ed in the data structure for p and we check to see whether either of the simplified inference rules for ∂ can be applied. This task is performed by *CheckInference*, which will add either $+\partial p$ or $-\partial p$, if justified, to the set S[3]. For example, *CheckInference* $(+\sigma p, S)$ will check whether $-\Delta{\sim}p$ and $-\sigma{\sim}p$ have already been established, so that $+\partial p$ may be inferred and added to S, and if $-\Delta{\sim}p$ has been established will add $-\partial{\sim}p$ to S. Similarly, *CheckInference*$(-\Delta p, S)$ will check whether $+\sigma{\sim}p$ and $-\sigma p$, so that $+\partial{\sim}p$ might be inferred, and will check $-\sigma p$, $+\Delta{\sim}p$ and $+\sigma{\sim}p$ to decide whether $-\partial p$ might be proved.

Execution of this algorithm can be understood as execution of the transition system of Section 3. $S \cup K$ combined with the set of recorded extended conclusions in the algorithm corresponds to C_i in the transition system. The deletions in the algorithm correspond to transitions 7, 8, 9, and 10. The add statements correspond to transitions 1 and 3. The record statements correspond to transitions 1, 2, 3, 2, and 4. These transitions are also reflected in the initialization of S. Finally, *CheckInference* embodies transitions 5 and 6.

The algorithm corresponds to a restricted form of the transition system where some sequences of transitions are disallowed. For example, the first case of Figure 1 requires that all transitions 7 and 8 involving p must occur as a block, uninterrupted

[3] Recall that defeasible logic will never infer both $+\partial p$ and $-\partial p$ since it is coherent (Billington, 1993).

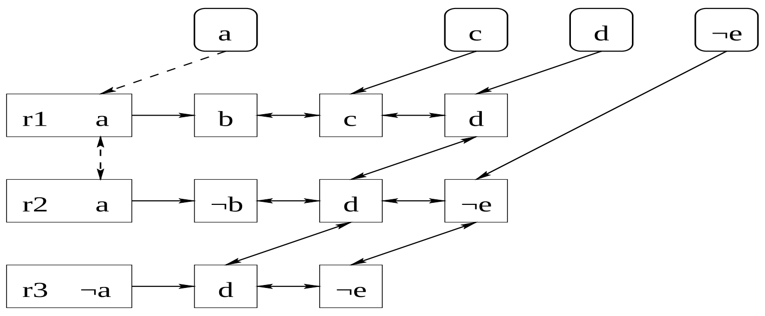

Fig. 2. Data structure for rules.

by any other uses of these transitions. Furthermore, the only possible other transitions in the block are transitions 1 and 2, when p is the last remaining literal in a rule, and transitions 5 and 6 when the recorded information about p triggers them. Note, however, that the algorithm does not disallow any transitions; it simply restricts the order in which transitions may occur. Thus the soundness and completeness of the transition system extends to soundness and completeness of the algorithm.

The key to an efficient implementation of this algorithm is the data structure used to represent the rules. It is exemplified (albeit incompletely) in Figure 2 for the theory

$$\begin{array}{rrr} r_1 : & b, c, d \Rightarrow & a \\ r_2 : & \neg b, d, \neg e \Rightarrow & a \\ r_3 : & d, \neg e \Rightarrow & \neg a \end{array}$$

Each rule body is represented as a doubly-linked list (horizontal arrows in Figure 2). Furthermore, for each literal p there are doubly-linked lists of the occurrences of p in the bodies of strict (respectively, defeasible) rules (diagonal arrows). For each literal p, there are doubly-linked lists of the strict (respectively, defeasible) rules for p (dashed arrows). Each literal occurrence has a link to the record for the rule it occurs in (not shown in Figure 2). As mentioned above, each literal p holds information showing which extended conclusions about p have been established, and there is also a link to the corresponding data structure for $\sim p$.

This data structure allows the deletion of literals and rules in time proportional to the number of literals deleted. Furthermore, we can detect in constant time whether a literal deleted was the only literal in that body, and whether a rule deleted with head h was the only strict (defeasible) rule for h. Each literal occurrence is deleted at most once, and the test for empty body is made at most once per deletion. Similarly, each rule is deleted at most once, and the test for no more rules is made once per deletion. The cost of a call to *CheckInference* is constant, since all checking relies on data that is immediately available, and at most three checks are needed per call. Thus the cost of the main part of the algorithm is $O(N)$, where N is the number of literal occurrences in D'.

Furthermore, the initialization of S is supported by the data structure and has cost proportional to the number of propositions in D'. Note that S can be implemented as a queue or a stack, since the only operations on S are to test for emptiness, add an element, or choose and delete an element. These operations require only constant time, and the number of operations is, in the worst case, proportional to the number of rules in D'. Since the initial transformations (Antoniou *et al.*, 2001) that produce D' are linear, the cost of the entire algorithm is linear in the size of D.

Theorem 5
The consequences of a defeasible theory D can be computed in $O(N)$ time, where N is the number of symbols in D.

This algorithm, when restricted to positive definite conclusions, is similar to the bottom-up linear algorithm for determining satisfiability of Horn clauses of Dowling & Gallier (1984) and Gallo & Urbani (1989). One difference is in the data structures: the Dowling-Gallier algorithm keeps a count of the number of atoms in the body of a rule, rather than keep track of the body. The latter results in greater memory usage, but allows us to reconstruct the residue of the computation: the simplified rules that remain. This residue is useful in understanding the behaviour of a theory, and can be useful in some applications of defeasible logic.

5 Discussion

There seem to be several inter-related features of defeasible logic that contribute to its linear complexity:

- *form of failure*: the form of failure-to-prove exhibited in the proof conditions for $-\Delta$ and $-\partial$ is closely related to Kunen's semantics for logic programs with negation-as-failure (Maher & Governatori, 1999). This form of failure does not include looping failures, and the propositional form is known to have linear complexity.
- *static and local nature of conflict*: when trying to derive $+\partial q$ or $-\partial q$ it is sufficient to look at rules for q and $\sim q$, since $\sim q$ is the only literal that conflicts with q. Thus the set of conflicting rules is unchanging and readily apparent. In logics where there is no bound on the number of conflicting literals and/or the conflicting rules cannot be known *a priori* (for example, plausible logic (Billington & Rock, 2001)) the corresponding inference rules for $+\partial$ and $-\partial$ can be expected to be more complex.
- *linear cost elimination of superiority relation*: defeaters can be handled as a variant of defeasible rules using the techniques of section 4. The presence of a superiority relation complicates these techniques greatly, although it is clear that there is a linear direct implementation of full defeasible logic by imitating the elimination transformation. The elimination of the superiority relation is based on a simple syntactic transformation (Antoniou *et al.*, 2001), but a similar transformation has been proposed for other logics (Kowalski & Toni, 1996) that have higher complexity. The success of such a simple transformation can be partly attributed to the static nature of the superiority relation and the static and local nature of conflicting literals, which do not change over the course of evaluation.

A significant non-factor in the complexity of defeasible logic is the team defeat aspect of the logic. This might be expected to create some difficulties since, conceptually, two teams of rules are pitted against each other, and this might give rise to combinatorial problems. However, reformulating the issue as one of competing/conflicting literals clarifies that there is a fixed number of pieces of information that are of interest, and each rule can contribute individually.

We can expect that similar logics, such as the variants discussed in (Antoniou *et al.*, 2001) and variants where strict rules are superior to defeasible rules (Nute, 1994), also have linear complexity and are amenable to the techniques used here, although the details will require careful verification. Similarly, well-founded defeasible logic (Maher & Governatori, 1999) can be expected to have quadratic complexity, since it employs the well-founded semantics (Van Gelder *et al.*, 1991) notion of failure, which has quadratic complexity.

6 Conclusion

We have shown that propositional defeasible logic has linear complexity. The algorithm that we have presented here has been implemented as the system Delores (Miller, 2000) and has undergone a preliminary experimental evaluation (Maher *et al.*, 2000). It executes simple, but large, theories of basic defeasible logic very quickly. The linear complexity of the algorithm is supported empirically.

However, the transformations used to convert an arbitrary defeasible theory to the appropriate form for the algorithm of Figure 1 impose a large constant factor on the cost of initializing S. Although the cause has not been pinpointed, it appears to be derived from the multiplication of rules and propositions during the transformations. It should be possible to improve performance by integrating the transformations of Antoniou *et al.* (2001) more tightly and/or abandoning the design goal (Antoniou *et al.*, 2001) of incrementality of the transformations (with respect to changes in the original defeasible theory).

Work is proceeding on improvements to the implementation of defeasible logic and the application of similar techniques to implement well-founded defeasible logic. We propose to use the implementation to support reasoning about regulations (Antoniou *et al.*, 1999).

Acknowledgements

Thanks to Grigoris Antoniou, David Billington, Guido Governatori and Andrew Rock for discussions on defeasible logic and its implementation. Thanks to Tristan Miller for discussion of the algorithm and his implementation of Delores. Also thanks to the anonymous referees for their careful reading and suggestions. This research was supported by the Australian Research Council.

References

Antoniou, G., Billington, D., Governatori, G. and Maher, M. J. (2000) A flexible framework for defeasible logics. *Proc. American National Conference on Artificial Intelligence (AAAI-00),* AAAI/MIT Press, pp. 405–410.

Antoniou, G., Billington, D., Governatori, G. and Maher, M. J. (2001) Representation results for defeasible logic. *ACM Transactions on Computational Logic*, **2**: 255–287.

Antoniou, G., Billington, D. and Maher, M. J. (1998) Normal forms for defeasible logic. *Proc. 1998 Joint International Conference and Symposium on Logic Programming*, MIT Press, pp. 160–174.

Antoniou, G., Billington, D. and Maher, M. J. (1999) On the analysis of regulations using defeasible rules. *Proc. 32nd Hawaii International Conference on Systems Science.*

Antoniou, G., Maher, M. J. and Billington, D. (2000) Defeasible logic versus logic programming without negation as failure. *Journal of Logic Programming*, **41**(1): 45–57.

Billington, D. (1993) Defeasible logic is stable. *Journal of Logic and Computation*, **3**: 370–400.

Billington, D., de Coster, K. and Nute, D. (1990) A modular translation from defeasible nets to defeasible logic. *Journal of Experimental and Theoretical Artificial Intelligence*, **2**: 151–177.

Billington, D. and Rock, A. (2001) Propositional plausible logic: introduction and implementation. *Studia Logica*, **67**(2): 243–269.

Cadoli, M. and Schaerf, M. (1993) A survey of complexity results for nonmonotonic logics. *Journal of Logic Programming*, **17**(2,3&4): 127–160.

Dimopoulos, Y. and Kakas, A. (1995) Logic programming without negation as failure. *Proc. 5th International Symposium on Logic Programming*, MIT Press, pp. 369–384.

Dowling, W. F. and Gallier, J. H. (1984) Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, **1**(3): 267–284.

Gallo, G. and Urbani, G. (1989) Algorithms for testing the satisfiability of propositional formulae. *Journal of Logic Programming*, **7**(1): 45–61.

Gottlob, G. (1992) Complexity results for nonmonotonic logics. *Journal of Logic and Computation*, **2**: 397–425.

Governatori, G. and Maher, M. J. (2000) An argumentation-theoretic characterization of defeasible logic. *Proc. European Conf. on Artificial Intelligence*, IOS Press, pp. 469–474.

Grosof, B. N. (1997) Prioritized conflict handling for logic programs. *Proc. Int. Logic Programming Symposium*, MIT Press, pp. 197–211.

Grosof, B. N. (1999) DIPLOMAT: Business Rules Interlingua and Conflict Handling, for E-Commerce Agent Applications (Overview of System Demonstration). *Proc. IJCAI-99 Workshop on Agent-mediated Electronic Commerce (AMEC-99).*

Horty, J. F. (1994) Some direct theories of nonmonotonic inheritance. In: Gabbay, D. M., Hogger, C. J. and Robinson, J. A. (editors), *Handbook of Logic in Artificial Intelligence and Logic Programming Vol. 3*, pp. 111–187, Oxford University Press.

Horty, J. F., Thomason, R. H. and Touretzky, D. (1987) A skeptical theory of inheritance in nonmonotonic semantic networks. *Proc. AAAI-87*, pp. 358–363.

Kowalski, R. and Toni, F. (1996). Abstract argumentation. *Artificial Intelligence and Law*, **4**: 275–296.

Maher, M. J. (2000) A denotational semantics for defeasible logic. *Proc. First International Conference on Computational Logic: LNAI 1861*, pp. 209–222. Springer-Verlag.

Maher, M. J., Antoniou, G. & Billington, D. (1998) A study of provability in defeasible logic. *Proc. Australian Joint Conference on Artificial Intelligence: LNAI 1502*, pp. 215–226. Springer-Verlag.

Maher, M. J. and Governatori, G. (1999) A semantic decomposition of defeasible logics. *Proc. American National Conference on Artificial Intelligence (AAAI-99),* pp. 299–305. AAAI/MIT Press.

Maher, M. J., Rock, A., Antoniou, G., Billington, D. and Miller, T. (2000) Efficient defeasible reasoning systems. *Proc. Int. Conf. on Tools in Artificial Intelligence*, pp. 384–392.

Miller, T. (2000) *DELORES User's Manual.*

Morgenstern, L. (1998) Inheritance comes of age: applying nonmonotonic techniques to problems in industry. *Artificial Intelligence*, **103**: 1–34.

Niemelä, I. (1999) Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, **25**(3-4): 241–273.

Nute, D. (1987) Defeasible reasoning. *Proc. 20th Hawaii International Conference on Systems Science*, pp. 470–477. IEEE Press.

Nute, D. (1994) Defeasible logic. In: Gabbay, D. M., Hogger, C. J. and Robinson, J. A. (editors), *Handbook of Logic in Artificial Intelligence and Logic Programming Vol. 3*, pp. 353–395. Oxford University Press.

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

Stein, L. A. (1992) Resolving ambiguity in nonmonotonic inheritance hierarchies. *Artificial Intelligence*, **55**: 259–310.

Stillman, J. (1992) The complexity of propositional default logics. *Proc. American National Conference on Artificial Intelligence (AAAI-92),* pp. 794–799. AAAI/MIT Press.

Van Gelder, A., Ross, K. A. and Schlipf, J. S. (1991) The well-founded semantics for general logic programs. *Journal of the ACM* **38**(3): 620–650.