A Family of Defeasible Reasoning Logics and its Implementation

G. Antoniou, D. Billington, G. Governatori, M.J. Maher and **A. Rock** [1]

Abstract. Defeasible reasoning is a direction in nonmonotonic reasoning that is based on the use of rules that may be defeated by other rules. It is a simple, but often more efficient approach than other nonmonotonic reasoning systems. This paper presents a family of defeasible reasoning formalisms built around Nute's defeasible logic. We describe the motivations of these formalisms and derive some basic properties and interrelationships. We also describe a query answering system that supports these formalisms and is available on the World Wide Web.

1 Introduction

Defeasible reasoning is a direction in nonmonotonic reasoning [13]. Defeasible logics were introduced and developed by Nute over several years [17]. 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 only when a rule supports it, and it has been demonstrated that no rule supports $\neg p$. 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.

In previous work we have studied a particular, "standard" defeasible logic, DL, and derived results concerning its representational properties [1] and proof theory [10], relationships to other formalisms [11, 3], and computational complexity [12].

Logics for knowledge representation and, in particular, nonmonotonic logics have developed greatly over the past 20 years. Many logics have been proposed, and a deeper understanding of the advantages and disadvantages of particular logics has been developed. There are also, finally, some indications that these logics can be usefully applied [15, 18].

Unfortunately, it appears that no single logic is appropriate in all situations, or for all purposes. History clearly indicates that while one logic may achieve desired results in some situations, in other situations the outcome is not as successful. This is, no doubt, one reason for the proliferation of non-monotonic logics.

Furthermore, even with a fixed syntax and a common motivating intuition, reasonable people can disagree on the semantics of the logic. This can be seen in the literature on semantics of logic programs with negation, for example, but the point was made more sharply by [21] who showed in several different ways a "clash of intuitions" for a simple language describing multiple inheritance with exceptions. So it appears that no single logic, with a fixed semantics, will be appropriate.

One way to address this problem is to develop logics that are "tunable" to the situation. That is, to develop a framework of logics in which an appropriate logic can be designed. In fact, families of approaches have emerged around the classical nonmonotonic systems of circumscription [14] and default logic [19]. In this paper we show how the original logic DL can be modified in different directions, to accommodate different intuitions. In particular, we show how team defeat (or absence of it) and ambiguity propagation (or blocking) can be accommodated in our framework. We also show that these ideas are orthogonal and may be combined as appropriate. A central result shows the relationship between the various approaches in the form of increasing deductive power.

As mentioned before one of the advantages of defeasible reasoning is that it was designed to support efficient reasoning. Therefore powerful implementations can be developed. In this paper we report on an implementation that supports query answering for defeasible logics. The experimental evaluation shows that it can deal with knowledge bases of the order of 100,000 defeasible rules with a response time of approx. 1 minute [12]. In this paper we will not focus on the experimental evaluation, but on other interesting features of the system. For example, it has been implemented in a functional programming language in a way that mirrors the inference conditions of the formalisms it supports. Therefore it can easily be extended to support further variants.

2 Defeasible Logic

We begin by outlining the constructs in defeasible logics [17]. We then define the inference rules of a particular defeasible logic DL that has received the most attention.

2.1 A Language of Defeasible Reasoning

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 containing free variables are interpreted as the set of their variable-free instances.

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:

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

[1] School of Computing and Information Technology, Griffith University, Nathan, QLD 4111, Australia.
{ga,db,guido,mjm,arock}@cit.gu.edu.au

Inference from facts and strict rules only is called *definite inference*. Facts and strict rules are intended to define relationships that are definitional in nature. Thus defeasible logics contain no mechanism for resolving inconsistencies in 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 $A \leadsto p$ 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

$$heavy(X) \leadsto \neg flies(X)$$

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 an acyclic 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

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

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

2.2 A Defeasible Logic

As an example of a defeasible logic, we consider the logic of [16], which has been investigated in [1, 10]. In this presentation we use the formulation given in [5]. We denote this logic by DL.

A *conclusion* of a defeasible theory D is a tagged literal. Conventionally [17, 5] there are four tags, so a conclusion has 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.

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 $\neg q$. 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. 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.

DL is closely related to several non-monotonic logics [2]. In particular, the "directly skeptical" semantics of non-monotonic inheritance networks [7] can be considered an instance of inference in DL once an appropriate superiority relation, derived from the topology of the network, is fixed [4]. DL is a conservative logic, in the sense of Wagner [22].

2.3 The Principle of Strong Negation

The purpose of the $-\Delta$ and $-\partial$ inference rules is to establish that it is not possible to prove a corresponding positive tagged literal. These rules are defined in such a way that all the possibilities for proving $+\partial q$ (for example) are explored and shown to fail before $-\partial q$ can be concluded. Thus conclusions with these tags are the outcome of a constructive proof that the corresponding positive conclusion cannot be obtained. As a result, there is a close relationship between the inference rules for $+\partial$ and $-\partial$, (and also between those for $+\Delta$ and $-\Delta$). The structure of the inference rules is the same, but the conditions are negated in some sense. We say that the inference rule for $+\partial$ ($-\partial$) is the *strong negation* of the inference rule for $-\partial$ ($+\partial$).

The strong negation of a formula is closely related to the function that simplifies a formula by moving all negations to an innermost position in the resulting formula. It is defined as follows.

$$\begin{array}{lcll}
sneg(A \wedge B) & = & sneg(A) \vee sneg(B) & \\
sneg(A \vee B) & = & sneg(A) \wedge sneg(B) & \\
sneg(\exists x\, A) & = & \forall x\; sneg(A) & \\
sneg(\forall x\, A) & = & \exists x\; sneg(A) & \\
sneg(+\partial p \in X) & = & -\partial p \in X & \\
sneg(-\partial p \in X) & = & +\partial p \in X & \\
sneg(\neg A) & = & \neg sneg(A) & \\
sneg(A) & = & \neg A & \text{if } A \text{ is a pure formula}
\end{array}$$

A pure formula is a formula that does not contain a tagged literal. Pairs of tags other than $+\partial, -\partial$ are treated in an analogous manner to $+\partial$ and $-\partial$. The strong negation of the applicability condition of an inference rule is a constructive approximation of the conditions where the rule is not applicable.

We are led to consider the following Principle of Strong Negation:

For each pair of tags such as $+\partial, -\partial$, the inference rule for $+\partial$ should be the strong negation of the inference rule of $-\partial$ (and vice versa).

Clearly DL satisfies this principle, as do all the logics we will present in the following. On the other hand, in Nute's framework [17] logics may violate it.

3 New Variants of Defeasible Logic

We now develop several variations of DL. Our interest here is not to develop definitive defeasible logics, but to demonstrate the flexibility of the framework. [11] have already defined an extension of DL to allow a failure operator in the body of rules without disturbing the semantics of DL on theories without this operator.

3.1 Removing Team Defeat

The defeasible logic we have considered so far incorporates the idea of *team defeat*. That is, an attack on a rule with head p by a rule with head $\sim p$ may be defeated by a *different* rule with head p (see inference rule $+\partial$). Even though the idea of team defeat is natural, it is worth noting that several related approaches, such as LPwNF [6] and most argumentation frameworks, do not adopt this idea.

It is easy to define a variant of DL that does not include team defeat. All we need to change in the inference condition $+\partial$ is clause (2.3.2) that should now look as follows:

(2.3.2) $r > s$

In other words, an attack on rule r by rule s can only be defended by r itself, in the sense that s is weaker than r. We use the tag ∂_{ntd} to refer to defeasible provability in this variant.

3.2 Ambiguity Propagation

We call a literal *ambiguous* if there is a chain of reasoning that supports the conclusion that p is true, another that supports the conclusion that $\neg p$ is true, and the superiority relation does not resolve this conflict. The following is a classic example of non-monotonic inheritance.

Example 1

$r_1 : \Rightarrow quaker$	$r_5 : \; republican \Rightarrow footballfan$
$r_2 : \Rightarrow republican$	$r_6 : \; pacifist \Rightarrow antimilitary$
$r_3 : \; quaker \Rightarrow pacifist$	$r_7 : \; footballfan \Rightarrow \neg antimilitary$
$r_4 : \; republican \Rightarrow \neg pacifist$	

The priority relation is empty.

$pacifist$ is *ambiguous* since the combination of r_1 and r_3 support $pacifist$ and the combination of r_2 and r_4 support $\neg pacifist$. Similarly, $antimilitary$ is ambiguous.

In DL, the ambiguity of $pacifist$ results in the conclusions $-\partial pacifist$ and $-\partial\neg pacifist$. Since r_6 is consequently not applicable, DL concludes $+\partial\neg antimilitary$. This behaviour is called *ambiguity blocking*, since the ambiguity of $antimilitary$ has been blocked by the conclusion $-\partial pacifist$ and an unambiguous conclusion about $antimilitary$ has been drawn.

A preference for ambiguity blocking or ambiguity propagating behaviour is one of the properties of non-monotonic inheritance nets over which intuitions can clash [21]. Stein [20] argues that ambiguity blocking results in an unnatural pattern of conclusions in extensions of the above example. Ambiguity propagation results in fewer conclusions being drawn, which might make it preferable when the cost of an incorrect conclusion is high. For these reasons an ambiguity propagating version of DL is of interest.

In the example above, the ambiguity blocking behaviour of DL was caused by the following observation: $pacifist$ is ambiguous, so $-\partial pacifist$ is derivable. Then $-\partial pacifist$ is used to invalidate rule r_6, thus leaving the conclusion $\neg antimilitary$ without any counterarguments.

The solution to achieve ambiguity propagation behaviour is to separate the invalidation of a counterargument from the derivation of $-\partial$ tagged literals. We do so by introducing a third level of provability (besides definite and defeasible provability), called *support* and denoted by $\int$. Intuitively, a literal p is supported if there is a chain of reasoning that would lead us to conclude p in the absence of conflicts. In the example above, $pacifist$ is supported although $-\partial pacifist$ can be derived.

Thus in our modification of the $+\partial$ condition, a counterargument may be disregarded only if a literal in its body is known not to be supported ($-\int$), which is stronger than the previous condition that made it sufficient for a literal in the body not be defeasibly provable. First we modify the $+\partial$ condition (the $-\partial$ condition is modified accordingly following the Principle of Strong Negation).

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

Next we define the inference conditions for support. If we ignore the superiority relation we could define it simply as follows.

If $P(i+1) = +\int q$ then either
    $+\Delta q \in P(1..i)$ or
    $\exists r \in R_{sd}[q]\ \forall a \in A(r) : +\int a \in P(1..i)$

However, in situations where two conflicting rules can be applied and one rule is inferior to another, the inferior rule should not be counted as supporting its conclusion. Thus we refine the inference rule as follows ($-\int$ is defined accordingly):

```
$+\int$: If $P(i+1) = +\int q$ then either
    $+\Delta q \in P(1..i)$ or
    $\exists r \in R_{sd}[q]$ such that
        $\forall a \in A(r) : +\int a \in P(1..i)$, and
        $\forall s \in R[\sim q]$ either
            $\exists a \in A(s) : -\partial_{am} a \in P(1..i)$ or
            $r > s$
```

3.3 Combination

It is also worth noting that several features can be easily integrated in our framework. Here we show the design of an ambiguity propagating defeasible logic without team defeat. To do so we modify the inference conditions for $+\partial_{am}$ and $-\partial_{am}$ as follows[2]:

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

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

It is quite obvious that $+\partial_{am}$ is modified to $+\partial_{am,ntd}$ the same way that $+\partial$ was modified to $+\partial_{ntd}$. This observation underscores the orthogonality of the two concepts (team defeat, ambiguity propagation).

3.4 Properties and Relationships

First we show that all logics we have described above satisfy the basic property of *coherence*:

Theorem 1 *There is no defeasible theory T and literal q such that $T \vdash +\delta a$ and $T \vdash -\delta q$, where δ denotes any of the tags we have presented ($\Delta, \partial, \partial_{ntd}, \partial_{am}, \partial_{am,ntd}, \int$).*

Next we show that there exists a chain of increasing expressive power among several of the logics.

Theorem 2 $+\Delta \subset +\partial_{am,ntd} \subset +\partial_{am} \subset +\partial \subset +\int$.
For each inclusion there are defeasible theories in which the inclusion is strict.

[2] The inference conditions $+\int$ and $-\int$ must be modified to use $+\partial_{am,ntd}$ and $-\partial_{am,ntd}$ instead of $+\partial_{am}$ and $-\partial_{am}$ respectively.

We wish to point out that this result is deeper that it may look on the surface. For example, the relation $+\partial_{a,ntd} \subset +\partial_a$ appears trivial since the absence of team defeat makes the logic weaker. But notice that when the logic fails to prove a literal p and instead proves $-\partial p$, then that result may be used by the logic to prove another literal q that could not be proven if p were provable. In fact it is easily seen that defeasible provability in the original defeasible logic without team defeat is *not* weaker than defeasible provability with team defeat. Consider the following example:

Example 2

a, b, c, d	
$r_1 : a \Rightarrow p$	$r_4 : d \Rightarrow \neg p$
$r_2 : b \Rightarrow p$	$r_5 : p \Rightarrow \neg q$
$r_3 : c \Rightarrow \neg p$	$r_6 :\Rightarrow q$
$r_1 > r_3,\ r_2 > r_4$	

Then q is not defeasibly provable in DL (in the sense of $-\partial q$), but defeasibly provable in DL without team defeat ($+\partial_{am} q$).

As we have mentioned before, all inference conditions in our framework follow the Principle of Strong Negation. Together with the previous theorem it follows:

Theorem 3 $-\Delta \supset -\partial_{am,ntd} \supset -\partial_{am} \supset -\partial \supset -\int$.
For each inclusion there are defeasible theories in which the inclusion is strict.

4 Implementation

The formalisms illustrated above have been implemented in *Deimos*, a query answering system. It is a suite of tools that (i) supports our ongoing research into defeasible reasoning; and (ii) is provided for public use to apply defeasible reasoning to practical problems. In accordance with these goals, *Deimos* was designed to satisfy, in decreasing order of significance, the following requirements: (1) correctness; (2) traceability; (3) flexibility and maintability; and (4) efficiency. With these goals in mind, Haskell was chosen as the implementation language.

The most important part of the system is the prover, which attempts to prove a literal at a selected level of proof (definite, defeasible, support) in the appropriate variant (with or without team defeat and ambiguity propagation), using backward chaining. This must be traceable to permit the verification of the inference conditions (important when adding new inference conditions, for student learning, and as a means of raising the confidence of external users in the system). Haskell's monadic I/O system allows us to provide a traceable execution while letting the code almost transparently represent the inference conditions (that we presented earlier in the paper). As an illustration of the transparency, we show the representation of the $+\partial$ inference condition in Haskell (where appropriate combinators `&&&`, `|||`, `fA`, `tE` are used):

```
(|--) t (Plus PS_d q) (|-)
= t |- Plus PS_D q |||
tE (rsdq t q) (\r -> fA (ants t r) (\a -> t |- Plus PS_d a)) &&&
t |- Minus PS_D (neg q) &&&
fA (rq t (neg q)) (\s ->
     tE (ants t s) (\a -> t |- Minus PS_d a) |||
     tE (rsdq t q) (\u ->
          fA (ants t u) (\a -> t |- Plus PS_d a) &&& beats t u s))
```

A big advantage of *Deimos* is precisely the one-to-one correspondence between the inference conditions and their representation as a Haskell expression. As such it provides great flexibility as a research tool because it is easy to verify and easy to modify as new inference conditions are developed for new defeasible reasoning systems. In fact the design of the defeasible reasoning logics described above went hand-in-hand with their implementation. Once the basic framework was put into place, it took us only minutes to implement each of the variants in *Deimos*. That way we were immediately able to test our intuitions on the behaviour of the logics and the relationships among the logics, before formally proving them.

The `prove` function prints a trace of all sub-goals. Some state information is also threaded through the evaluation of the inference conditions, including a history of all the sub-goals so far encountered and their corresponding proof status. The history enables loop detection and saves re-evaluating previously encoutnered goals. The loop checking guarantees that the prover terminates for all queries. It should be noted that the history and loop checking functions may be turned off, if appropriate.

Deimos is available as a command line tool which parses a defeasible theory and attempts to prove conclusions, printing a trace if required.

An alternative user interface is provided by a CGI version[3], which is also written in Haskell. This web-accessible version is the preferred interface for most users, and has links to explanatory information such as the syntax for all inputs, help, and a complete user's manual including the complete code. It also comes with a library of pre-prepared defeasible theories (we are in the process of enhancing this library). The present system now consists of about 4000 lines of Haskell code.

5 Current and Future Work

In this paper we showed that a family of defeasible reasoning formalisms can be built around defeasible logic. These formalisms allow one to tune the logic according to the needs of some particular application and according to one's intuitions. We also described an implementation of the formalisms, that is designed to be flexible and easily extendable to include further variants. A main feature of the implementation is its transparent representation of the inference conditions; therefore it has been instrumental even in our conceptual considerations.

Apart from the variants presented in this paper, we are currently working on further formalisms. We have developed a well-founded defeasible logic, and are in the process of incorporating *dynamic priorities* into the framework. We are also working on a full propositional version of defeasible logic, called plausible logic, and on the addition of variables to *Deimos*. Finally we are studying relationships to systems of argumentation, and are working on applications in the areas of modelling regulations and business rules, and the legal domain. We believe that defeasible reasoning is promising to be used in practical applications: on one hand it is simple (rule-based) and sufficiently efficient, and on the other hand it is declarative with all the associated advantages.

Acknowledgements

This research was supported by the Australia Research Council. under Large Grant No. A49803544.

[3] `www.cit.gu.edu.au/∼arock/defeasible/Defeasible.cgi`

REFERENCES

[1] G. Antoniou, D. Billington and M.J. Maher. 1998. Normal Forms for Defeasible Logic. In *Proc. Joint International Conference and Symposium on Logic Programming*, J. Jaffar (Ed.), 160–174. MIT Press.

[2] G. Antoniou, M.J. Maher, and D. Billington. 1999. Defeasible Logic versus Logic Programming without Negation as Failure. submitted.

[3] Defeasible Logic versus Logic Programming without Negation as Failure. *Journal of Logic Programming* 41,1 (2000): 45–57.

[4] D. Billington, K. de Coster and D. Nute. 1990. A Modular Translation from Defeasible Nets to Defeasible Logic. *Journal of Experimental and Theoretical Artificial Intelligence* 2: 151–177.

[5] D. Billington. 1993. Defeasible Logic is Stable. *Journal of Logic and Computation* 3: 370–400.

[6] Y. Dimopoulos and A. Kakas. 1995. Logic Programming without Negation as Failure. In *Proc. ICLP-95*, MIT Press.

[7] J.F. Horty, R.H. Thomason and D. Touretzky. 1987. A Skeptical Theory of Inheritance in Nonmonotonic Semantic Networks. In *Proc. AAAI-87*, 358–363.

[8] K. Kunen. 1987. Negation in Logic Programming. *Journal of Logic Programming* 4: 289–308.

[9] J. W. Lloyd and R. W. Topor. 1984. Making Prolog more Expressive. *Journal of Logic Programming* 1(3): 225–240.

[10] M.J. Maher, G. Antoniou and D. Billington. 1998. A Study of Provability in Defeasible Logic. In *Proc. Australian Joint Conference on Artificial Intelligence*, 215–226, LNAI 1502, Springer.

[11] M.J. Maher and G. Governatori. 1999. A Semantic Decomposition of Defeasible Logics. *Proc. American National Conference on Artificial Intelligence (AAAI-99),* 299–306.

[12] M.J. Maher, A. Rock, G. Antoniou, D. Billington and T. Miller. Efficient Defeasible Reasoning Systems. Submitted to the *1st International Conference on Computational Logic*.

[13] V. Marek and M. Truszczynski. *Nonmonotonic Logic*, Springer 1993.

[14] J. NcCarthy. Circumscription – A Form of Non-Monotonic Reasoning. *Artificial Intelligence* 13 (1980): 27–39.

[15] L. Morgenstern. 1998. Inheritance Comes of Age: Applying Nonmonotonic Techniques to Problems in Industry. *Artificial Intelligence,* 103, 1–34.

[16] D. Nute. 1987. Defeasible Reasoning. In *Proc. 20th Hawaii International Conference on Systems Science*, IEEE Press, 470–477.

[17] D. Nute. 1994. 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, 353–395.

[18] H. Prakken. 1997. *Logical Tools for Modelling Legal Argument: A Study of Defeasible Reasoning in Law.* Kluwer Academic Publishers.

[19] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence* 13(1980): 81–132.

[20] L.A. Stein. 1992. Resolving Ambiguity in Nonmonotonic Inheritance Hierarchies. *Artificial Intelligence* 55: 259–310.

[21] D.D. Touretzky, J.F. Horty and R.H. Thomason. 1987. A Clash of Intuitions: The Current State of Nonmonotonic Multiple Inheritance Systems. In *Proc. IJCAI-87*, 476–482, Morgan Kaufmann, 1987.

[22] G. Wagner. 1991. Ex Contradictione Nihil Sequitur. In *Proc. IJCAI-91*, 538–546 , Morgan Kaufmann.