Abstract Argumentation

ROBERT A. KOWALSKI and FRANCESCA TONI
Department of Computing Imperial College of Science, Technology and Medicine 180 Queen's Gate, London SW7 2BZ, UK, {rak,ft}@doc.ic.ac.uk

5 September 1996

Abstract. In this paper we explore the thesis that the role of argumentation in practical reasoning in general and legal reasoning in particular is to justify the use of defeasible rules to derive a conclusion in preference to other defeasible rules to derive a conflicting conclusion. The defeasibility of rules is expressed by means of non-provability claims as additional conditions of the rules.

We outline an abstract approach to defeasible reasoning and argumentation which includes many existing formalisms, including default logic, extended logic programming, non-monotonic modal logic and auto-epistemic logic, as special cases. We show, in particular, that the "admissibility" semantics for all these formalisms has a natural argumentation-theoretic interpretation and proof procedure, which seem to correspond well with informal argumentation.

In the admissibility semantics there is only one way for one argument to attack another, namely by undermining one of its non-provability claims. In this paper, we show how other kinds of attack between arguments, specifically how rebuttal and priority attacks, can be reduced to undermining non-provability claims.

1. Introduction

The purpose of this paper is to outline a formal theory of argumentation, which promises to have relevance for practical reasoning in general and for legal reasoning in particular. We will argue that the role of argumentation is to justify the use of certain defeasible rules deriving a conclusion in preference to other defeasible rules deriving conflicting conclusions.

We explore the thesis that defeasible reasoning with rules of the form

P <u>if</u> Q.

can be understood as "exact" reasoning with rules of the form

P <u>if</u> Q <u>and</u> S <u>can not be shown</u>.

containing one or more defeasible "non-provability claims" of the form

S <u>can not be shown</u>, for some sentence S.

In many cases, the sentence S is the "contrary", not P, of the conclusion, P, of the rule. In other cases, S may state that the rule itself is "defeated" by another rule.

With this understanding of defeasibility, argumentation is a dialectic process whereby a proponent presents an "exact" argument of a conclusion, which is based, however, upon defeasible non-provability claims of the form S can not be shown. Such a claim, and the argument it helps to support, can be defeated if an opponent undermines the claim by presenting an argument for S. Like the proponent, the opponent can also base her/his argument upon non-provability claims.

The argumentation process can be viewed, therefore, as a game in which the proponent moves first. In the "credulous" version of the game, which we study in this paper, by moving first, the proponent has the advantage of being able to use his/her previously used claims of non-provability to defeat the opponent's counter-claims. Such credulous argumentation can be used to justify more than one point of view. If the opponent moves first, the opponent can use her/his previously used claims to defeat the proponent. In the "sceptical" version of the game, which we do not explore in this paper, the proponent can use only undisputed claims of non-provability to defeat the opponent's counter-claims.

A similar view of defeasible reasoning and argumentation has been put forward by several authors, e.g. [17, 18, 10]. Our approach differs, however, in several respects. The most important of these are:

1. Our approach to argumentation focuses on the acceptability of the non-provability claims of an argument, rather than on the acceptability either of the argument or of the conclusion of the argument. This concentrates attention, therefore, on the contentious parts of the argument and avoids being distracted by those parts of the argument that are beyond dispute.
2. We reduce all forms of defeasibility to that of non-provability claims. As a consequence, the only way to defeat an argument is by undermining one of its claims, namely by presenting an argument for a sentence S which is claimed to be unprovable. Indirect defeat (also called "rebutting" [17, 18] or "reductio-ad-absurdum" [6]), showing that a rule

   P if Q.

   leads to contradiction, is transformed into undermining defeat, by rewriting the rule in the "exact" form

   P if Q and not P can not be shown.

   making explicit in the conditions of the rule that the contrary, not P, of its conclusion, P, can not be shown.

3. Our semantics is credulous rather than sceptical. A sceptical semantics can be obtained from credulous semantics by regarding a conclusion as justified if and only if it can be derived from any credulously "acceptable" set of claims.
4. Our methodology deals with priorities between rules by introducing explicit rule names into the language and adding conditions to the rules expressing that they are not defeated by other higher priority rules. This approach avoids the need to deal with priorities in the semantics.
5. Our approach does not require the introduction of a new language or of a new semantics. On the contrary, we base our formalisation upon a variant of an abstract approach to defeasible reasoning [2] which has been shown to include many existing formalisms, including (extended) logic programming, default logic [20], non-monotonic modal logic [15] and autoepistemic logic [16] as special cases. Thus the approach taken in this paper is abstract and can be formalised in any one of these and other formalisms. Although all these formalisms use the "stability" semantics, we argue that the "admissibility" semantics is more adequate.

The paper is organised as follows. In section 2, we illustrate our approach by means of examples, which are used elsewhere in the paper. In section 3, we describe the abstract framework for defeasible reasoning, and present the stability semantics. In section 4, we show how default logic, extended logic programming, non-monotonic modal logic, and auto-epistemic logic can all be considered special cases of the framework. Thus our approach to defeasible reasoning and argumentation can be formulated in any one of these and other formalisms for default reasoning. In section 5, we define the admissibility semantics and its argumentation proof procedure. In section 6, we show how our approach reduces rebutting attacks to undermining attacks. In section 7, we show how the approach reduces priorities to non- provability conditions. In section 8, we conclude.

We assume the reader is already familiar with the general subject of argumentation and its relevance to legal reasoning. In particular, we recommend the article by Prakken and Sartor in this issue [19] for an overview of related work.

The framework we use in this paper is a variant of one we have developed elsewhere [2]. The reader will find formal definitions and results in [2]. The main technical contribution of this paper, therefore, is the methodology we present for eliminating rebuttal attacks and priorities from the semantics. The main general contribution is our use of the methodology for argumentation which is abstract and language-independent.

2. Examples

EXAMPLE 2.1. Consider the following statement of the principle that, by default, a person accused of a crime should be assumed innocent unless the person can be shown not innocent.

(1) *A person is innocent of a crime*
 if *the person is accused of committing the crime*
 and it can not be shown that *the person is* not *innocent of the crime.* [1]

The inclusion in the rule of the explicit non-provability condition transforms a defeasible rule

(1′) *A person is innocent of a crime*
 if *the person is accused of committing the crime.*

which might be subject to argument, into an "exact" rule, which is beyond dispute. Any dispute about the defeasible rule is transformed, instead, into a dispute about the non-provability condition.

In contrast, the "exact" rule

(2) *A person is* not *innocent of a crime*
 if *the person was observed committing the crime.*

does not contain any non-provability conditions and consequently does not represent a defeasible rule.

Suppose now that we are given the fact

(3) *John is accused of theft.*

We can justify the conclusion that John is innocent by the following argument.

Proponent: *John is innocent of theft,*
 because (although *John is accused of theft*)
 it can not be shown that *John is* not *innocent of theft.*

The proponent's claim, it can not be shown that *John is* not *innocent of theft*, is upheld, because the only way to defeat it, by using rule (2), fails to apply, given the lack of any "fact" ("exact" rule) recording an observation of his having committed the crime. The claim can be defeated, however, if such information becomes available at a later time. It is because of this that such logic is often said to be "non-monotonic", in contrast to "monotonic" logic, where once a conclusion is established it continues to hold no matter what further information is added.

Although the two rules, (1) and (2) above, are both "exact", the inclusion of the extra condition in (1) can be understood as giving the defeasible rule, (1′), that (1) represents, a lower priority than (2). We shall argue more generally, in section 7, that priorities involving defeasible rules can be represented by explicit non-provability conditions of "exact" rules.

EXAMPLE 2.2. Consider the two defeasible rules

A person inherits an estate
    if *he/she is the beneficiary of a valid will.*
A person does not *inherit an estate*
    if *he/she murders the owner of the estate.*

and the facts

John is the beneficiary of Henry's valid will.
John has murdered Henry.

Everything else being equal, in a credulous approach to defeasible reasoning, there are two equally "acceptable" arguments: one concluding that John inherits Henry's estate because he is the beneficiary of Henry's valid will; the other that he does not inherit the estate because he has murdered Henry. In a sceptical approach (which we do not discuss in this paper, but we do discuss in [2]) neither conclusion holds.

In this paper, we will consider two kinds of "acceptable" arguments, namely "stable" (section 3) and "admissible" (section 5). The two arguments above, one concluding that John inherits Henry's estate and the other concluding that he does not inherit the estate, are "acceptable" in both senses.

In our analysis of defeasibility and argumentation, we justify the credulous reasoning of this example in the following way:

To say that the first two rules are defeasible is to say that they have implicit conditions to the effect that the contrary of their conclusions can not be shown. We make these conditions explicit by rewriting the rules in the "exact" form

A person inherits an estate
    if *he/she is the beneficiary of a valid will*
    and it can not be shown that *the person does* not *inherit the estate.*
A person does not *inherit an estate*
    if *he/she murders the owner of the estate*
    and it can not be shown that *he/she inherits the estate.*

We can now justify the conclusion that John inherits the estate by the following dialogue game.

Proponent: *John inherits the estate*,
because *John is the beneficiary of a valid will*
and because
it can not be shown that *John does not inherit the estate.*

Opponent: On the contrary,
it can be shown that *John does not inherit the estate*
because *John murdered the owner of the estate*
and because
it can not be shown that *John inherits the estate.*

Proponent: But, on the contrary,
it can be shown that *John inherits the estate*,
as I have already shown.

The rules of the game are designed to be liberal about the arguments that are "admissible". An argument is "admissible" if its non-provability claims can be defended against any counter-claims of an opponent. By being the first player, the proponent has the advantage of being able to use his/her earlier claims to defeat the opponent's counter-claims. This enables the proponent, in this example, to have the last word and to win the argument. If the opponent is allowed to move first, then the opponent's argument will be equally "admissible".

It is also possible to design a conservative version of the game, which is sceptical about what conclusions can be established.

EXAMPLE 2.3. (Adapted from [18])
Consider the following general principle of European Community law:

a product can be sold in a country
if *the country is part of the EC*
and *the product can be sold in another country*
and *the other country is part of the EC*
and it can not be shown that
the selling of the product in the country endangers public health
and it can not be shown that
the selling of the product in the country prejudices the consumer.

This principle potentially conflicts with the following rule of Italian law:

a product can not *be sold in Italy*

    <u>if</u> *the product is called "pasta"*
    <u>and</u> *the product is* <u>not</u> *made of durum wheat.*

The potential conflict becomes real, when we are given only the following facts:

Italy is part of the EC.
BDR is part of the EC.
alpha is sold in BDR.
alpha is called "pasta".
alpha is <u>not</u> *made of durum wheat.*

If the two rules above are understood as "exact" rules, then it is possible to derive the inconsistent conclusion that *alpha* can be sold in Italy and *alpha* can not be sold in Italy. This derivation is based upon the two non-provability claims of the first rule, neither of which can be undermined, using the given rules and facts.

However, if the two rules are understood as defeasible, then in our methodology for representing defeasibility we would transform them into "exact" rules by adding extra non-provability conditions to both, rewriting them as:

a product can be sold in a country
    <u>if</u> *the country is part of the EC*
    <u>and</u> *the product can be sold in another country*
    <u>and</u> *the other country is part of the EC*
    <u>and</u> <u>it can not be shown that</u>
        the selling of the product in the country endangers public health
    <u>and</u> <u>it can not be shown that</u>
        the selling of the product in the country prejudices the consumer
    <u>and</u> <u>it can not be shown that</u>
        the product can <u>not</u> *be sold in the country.*
a product can <u>not</u> *be sold in Italy*
    <u>if</u> *the product is called "pasta"*
    <u>and</u> *the product is* <u>not</u> *made of durum wheat*
    <u>and</u> <u>it can not be shown that</u> *the product can be sold in the country.*

As in example 2.2, it would now be possible to construct two separate "admissible" arguments, each of which undermines the other, rather than one "admissible" argument with an inconsistent conclusion.

However, if, as is the case, we want to give higher priority to European Community law than to Italian law, then we would simply retain the original formulation of the Italian law, as an "exact" rule, without a non-provability condition, following the lead of example 2.1.

Thus, by adding appropriate non-provability conditions to rules, expressing that the contrary of their conclusions can not be shown, we obtain the same result as Prakken and Sartor [18], who, in addition to using undermining attacks, also employ indirect rebutting attacks and deal with priorities in the semantics. We will see later in section 7 that the treatment of priorities in this example can be seen as an optimised version of a more general approach that can handle more complex priorities, like those of Prakken and Sartor [18], which can be defined by means of rules. This approach is illustrated by the following example.

EXAMPLE 2.4. (Adapted from [11])
Consider the principles:

r_1: Except as provided for by r_2, all thieves should be punished.

r_2: Except as provided for by r_3, thieves who are minors should be rehabilitated and not punished.

r_3: Any thief who is violent should be punished.

These can be represented by the defeasible rules

r_1: *A person should be punished*
 <u>if</u> *the person is a thief.*
r_2: *A person should* <u>not</u> *be punished*
 <u>if</u> *the person is a thief*
 <u>and</u> *the person is a minor.*
r_3: *A person should be punished*
 <u>if</u> *the person is a thief*
 <u>and</u> *the person is violent.*

where rule r_3 has higher priority than rule r_2, and r_2 higher priority than rule r_1 ($r_3 > r_2$ and $r_2 > r_1$).

Given the facts

f_1: *John is a thief.*
f_2: *John is a minor.*

rules r_1 and r_2 conflict. Intuitively, the information $r_2 > r_1$ can be used to derive the conclusion *John should* <u>not</u> *be punished* of the second rule, in preference to the conclusion of the first. If we are given the additional fact

f_3: *John is violent.*

then rules r_1, r_2 and r_3 now conflict. Intuitively, the information $r_3 > r_2$ can be used to derive the conclusion *John should be punished* of the third rule.

In the previous example 2.3 we used a simple method to represent defeasible rules as "exact" rules while taking into account the priorities between them. The method consists of adding non-provability conditions, that the contrary of their conclusions can not be shown, to defeasible rules with lower priority, without adding similar conditions to the higher priority rules. This method can not be used in this example, since rule r_2 has higher priority than rule r_1 but lower priority than rule r_3.

In such an example, we need to distinguish the different ways a conclusion can be established, and add non-provability conditions to defeasible rules with lower priority stating that no higher priority rule applies. In this way, using this more refined methodology, we obtain the following representation of the example.

A person should be punished
 if *the person is a thief*
 and it can not be shown that r_1 *is defeated for the person.*
r_1 *is defeated for a person*
 if *the person is a thief*
 and *the person is a minor*
 and it can not be shown that r_2 *is defeated for the person.*

A person should not *be punished*
 if *the person is a thief*
 and *the person is a minor*
 and it can not be shown that r_2 *is defeated for the person.*
r_2 *is defeated for a person*
 if *the person is a thief*
 and *the person is violent*
 and it can not be shown that r_3 *is defeated for the person.*

A person should be punished
 if *the person is a thief*
 and *the person is violent*
 and it can not be shown that r_3 *is defeated for the person.*

Now, given the facts f_1 and f_2, there is only one "admissible" argument for the conclusion *John should* not *be punished*, supported by the non-provability claim it can not be shown that r_2 *is defeated for John*. However, given the facts f_1, f_2 and f_3, there is only one "admissible" argument for the conclusion *John should be punished*, supported by the non-provability claim it can not be shown that r_3 *is defeated for John*. This argument undermines the conflicting claim it can not be shown that r_2 *is defeated for John*, supporting the conclusion that *John should* not *be punished*.

3. An abstract framework for representing defeasibilty

All the examples above have been formulated in a form of English which can be represented directly in many existing (and also in many presently uninvestigated) formalisms for default reasoning. Indeed, to formalise such examples, we need only to choose a logic of "exact" (i.e. monotonic, non-defeasible) reasoning (such as classical first-order logic) and extend it, if necessary, by adding a non-provability operator. We then need to define a semantics and proof theory for the extended language, which appropriately interprets the intended meaning of the non-provability operator. More precisely

1. We assume an underlying **monotonic** (or **exact**) **logic** with a logical connective representing implication (to which modus ponens applies) and a universal quantifier (to which universal quantifier elimination applies).
2. We extend the language of the underlying logic, if necessary, by means of a **non-provability operator**, to express statements of the form

   S <u>can not be shown</u>

   where S names a sentence of the language of the underlying logic. We restrict the occurrence of these non-provability statements to conditions of implications.
3. Defeasible rules can then be represented in the extended language by transforming them into (exact) universally quantified implications with non-provability conditions.
4. Defeasible reasoning from a set T (called a **theory**) of sentences in the extended language, is reduced to exact reasoning in the underlying logic, from an extended set of sentences $T \cup N$, where N is an "acceptable" set of non-provability claims of the form

   S <u>can not be shown</u>.

Since, in most cases [2], the underlying language does not contain non-provability formulae, we have to explain how, in such cases, the underlying logic can be used to derive conclusions from the extended theory $T \cup N$. For this purpose, we assume that sentences in N are treated as atomic propositions. We also assume that instantiation can be applied to universally quantified implications representing defeasible rules, obtaining instantiated implications with atomic non-provability conditions. Modus ponens can then be applied to such instantiated implications and sentences in N, obtaining sentences of the original underlying language.

It remains to specify, given a theory T, when a set of non-provability claims N is "acceptable". Almost all existing logics for default reasoning solve this problem in the same way, by means of a requirement that N be "stable", i.e.

A candidate set N of sentences of the form S can not be shown is said to be **stable** if and only if for every S in the underlying language,
S can not be shown is in N if and only if
S is not derivable from $T \cup N$ in the underlying logic.

Different logics for default reasoning formulate this stability requirement in different ways, which are all, nonetheless, equivalent [2]. [3]

Despite its being the dominant semantics for default reasoning, stability semantics has a number of drawbacks, the most obvious being that to verify the acceptability of a claim S can not be shown necessitates determining an entire stable set N to show that S is not derivable from $T \cup N$. This is computationally infeasible in most interesting cases. There are also cases where a stable set does not exist because of a local paradoxical sentences in T, but where an intuitively acceptable argument can be constructed from other parts of T which are not affected by the paradox.

Consider again the theory T consisting of the set of sentences (1), (2) and (3) of example 2.1. Intuitively, the set N consisting of the single sentence

(4) *John is* not *innocent of theft* can not be shown

is acceptable, because there is no way to show, given only T (and possibly other competing non-provability claims) that *John was observed committing theft.*

The conclusion that *John is innocent of theft* then follows from $T \cup N$ in the underlying logic, by instantiating (1), obtaining the variable-free implication

John is innocent of theft
if *John is accused of theft*
and it can not be shown that *John is* not *innocent of theft.*

and then applying modus ponens, using (3) and (4).

Intuitively, in this example it is unnecessary to consider the set of all sentences unprovable from $T \cup N$ to be convinced that the single sentence in N is unprovable. Similarly, even if T is extended to a much larger set of sentences T', the same argument presented above continues to hold, provided T' does not contain any new sentences which can be used to derive either that *John was observed committing the crime* or to derive more directly that *John is* not *innocent of theft*. Even a paradoxical sentence of the form

$$P \ \underline{\text{if}} \ P \ \underline{\text{can not be shown}}$$

in T' need not affect the correctness of the argument.

In section 5, we will show that these drawbacks of the stability semantics can be overcome by means of the more liberal "admissibility" semantics [5], which has an argumentation-theoretic interpretation, as illustrated informally already in examples 2.1 and 2.2. The "admissibility" semantics is compatible with stability semantics, in the sense that whenever a stable extension exists it is "admissible". Moreover, in many cases an "admissible" set can be extended to a stable set [2].

First, we show how a number of existing formalisms for default reasoning can be understood as instances of the abstract framework.

4. Special instances

4.1. Default logic

1. The underlying logic is first-order classical logic, augmented with domain-specific inference rules. These inference rules can equivalently be formulated as sentences of the form

   $$P \ \underline{\text{if}} \ Q_1 \ \underline{\text{and}} \ \ldots \underline{\text{and}} \ Q_n.$$

   where P, Q_1, ..., Q_n are all first-order formulae of the underlying language, $n \geq 0$, and if is a new implication sign, for which the inference rule of modus ponens applies. Notice that in this formulation there are two logical connectives for implication, the new implication symbol and the material implication of classical logic.

   Any variables in these new implications which are not explicitly quantified are implicitly universally quantified, with as scope the entire implication. The rule of instantiation applies to such variables.
2. Non-provability is expressed by means of a logical operator, M, where the intended meaning of $M\,P$, where P is a sentence of first-order logic, is that the contrary of P, namely not P, can not be shown. $M\,P$ can be interpreted equivalently as expressing that P is consistent.
3. Thus defeasible rules are expressed in default logic by translating them into exact sentences of the form

   $$P \ \underline{\text{if}} \ Q_1 \ \underline{\text{and}} \ \ldots \underline{\text{and}} \ Q_n \ \underline{\text{and}} \ M\,P_1 \ \underline{\text{and}} \ \ldots \underline{\text{and}} \ M\,P_m.$$

   where $n \geq 0, m > 0$ and apparently free variables are implicitly universally quantified. Statements of the form $M\,P$ are interpreted as atoms in the underlying first-order language.

4. The standard semantics of default logic [20] has been shown [2] to be a special case of the stable semantics defined in the previous section for abstract frameworks in general.

4.2. Extended logic programming

1. The underlying language consists of implications of the form

$$P \ \underline{\text{if}}\ Q_1 \ \underline{\text{and}} \ldots \underline{\text{and}}\ Q_n.$$

where P, Q_1, …, Q_n are literals, i.e. atomic formulas A or the classical negation $\underline{\text{not}}\ A$ of an atomic formula, $n \geq 0$ and $\underline{\text{if}}$ is an implication sign. Variables in implications are implicitly universally quantified with scope the entire implication. The only inference rules are modus ponens for "$\underline{\text{if}}$" and instantiation for the implicit universal quantifiers.

2. Non-provability is expressed by means of a logical operator, $\underline{\text{naf}}$ (negation as failure), where the intended meaning of $\underline{\text{naf}}\ P$, where P is a literal, is that P can not be shown. Therefore, $\underline{\text{naf}}\ \underline{\text{not}}\ P$ expresses that the contrary of P can not be shown, i.e. that P is consistent.

3. Thus defeasible rules are expressed in extended logic programming by translating them into exact sentences of the form

$$P \ \underline{\text{if}}\ Q_1 \ \underline{\text{and}} \ldots \underline{\text{and}}\ Q_n \ \underline{\text{and}}\ \underline{\text{naf}}\ P_1 \ \underline{\text{and}} \ldots \underline{\text{and}}\ \underline{\text{naf}}\ P_m.$$

where $n \geq 0, m > 0$ and apparently free variables are implicitly universally quantified. Statements of the form $\underline{\text{naf}}\ P$ are interpreted as atoms in the underlying language.

4. The answer set semantics of extended logic programming [9] has been shown [2] to be a special case of the abstract stable semantics, where the theory corresponding to an extended logic program is understood as implicitly containing all implications of the form

$$P \ \underline{\text{if}}\ Q \ \underline{\text{and}}\ \underline{\text{not}}\ Q.$$

where P is any literal and Q is any atomic formula of the underlying language.

Normal logic programs can be understood as extended logic programs without any occurrence of negative literals of the form $\underline{\text{not}}\ P$. Normal logic programming is the minimal language that is an instance of the abstract framework. The stable model semantics of normal logic programming [8] has been shown [2] to be a special case of the abstract stable semantics.

As argued in [13, 21, 18], extended logic programming is better suited for legal knowledge representation than normal logic programming, due to the presence, in extended logic programming, of both "classical" negation ($\underline{\text{not}}$) and non-provability ($\underline{\text{naf}}$).

4.3. Non-monotonic modal logic and autoepistemic logic

1. In non-monotonic modal logic, the underlying logic can be any modal logic with a modal operator, L, which can be interpreted as representing provability. Clearly, some modal logics are not appropriate for this purpose. However, all modal logics incorporate an inference rule, called necessitation, which sanctions deriving $L\,P$ from P.
   One of the most frequently advocated modal logics for this purpose is K45, which contains the axiom schemata

   (K) $L\,\psi \;\underline{\text{if}}\; (L\,(\psi \;\underline{\text{if}}\; \phi) \;\underline{\text{and}}\; L\,\phi).$
   (4) $L\,L\,\phi \;\underline{\text{if}}\; L\,\phi.$
   (5) $L \;\underline{\text{not}}\; L\,\phi \;\underline{\text{if}}\; \underline{\text{not}}\; L\,\phi$

   where $\underline{\text{if}}$ is material implication [4]. Non-monotonic modal logic with K45 as underlying logic is equivalent to autoepistemic logic [22].
2. The underlying logic is already sufficiently expressive to represent non-provability. Non-provability is represented by means of formulae of the form $\underline{\text{not}}\; L\,P$, expressing that P can not be shown. Therefore, $\underline{\text{not}}\; L \;\underline{\text{not}}\; P$ expresses that the contrary of P can not be shown, i.e. that P is consistent. The first version of non-monotonic modal logic [15] was defined in terms of a modal operator M, standing for consistency. Obviously, the two modal operators, L and M, are interdefinable.
3. Defeasible rules are expressed in non-monotonic modal logic by translating them into exact sentences of the form

   $$P \;\underline{\text{if}}\; L\,Q_1 \;\underline{\text{and}}\; \ldots \underline{\text{and}}\; L\,Q_n \;\underline{\text{and}}\; \underline{\text{not}}\; L\,P_1 \;\underline{\text{and}}\; \ldots \underline{\text{and}}\; \underline{\text{not}}\; L\,P_m.$$

   prefixed by explicit universally quantified variables, where $n \geq 0$ and $m > 0$.
4. The fixed point semantics of non-monotonic modal logics [15] has been shown [2] to be a special case of the abstract stable semantics defined in section 3.

Arbitrary non-monotonic modal logics are more expressive than default logic and extended logic programming, because they allow the expression of complex modal sentences, with nested modal operators. However, Konolige [12] has shown that in propositional K45 any theory T can be reformulated equivalently as a theory in which all sentences have the form

$$P \ \underline{\text{if}}\ L\,Q_1\ \underline{\text{and}}\ \ldots \underline{\text{and}}\ L\,Q_n\ \underline{\text{and}}\ \underline{\text{not}}\ L\ P_1\ \underline{\text{and}}\ \ldots \underline{\text{and}}\ \underline{\text{not}}\ L\ P_m.$$

where the Q_i, P_j and P are non-modal, and $n, m \geq 0$.

5. Admissibility semantics and argumentation

Like the stability semantics, the admissibility semantics specifies that a conclusion C follows from a theory T if and only if C follows from $T \cup N$ in the underlying logic, where N is an appropriate set of non-provability claims. However, to be "admissible", N need only contain enough non-provability claims to derive C, supplemented by sufficiently many additional non-provability claims needed to defend itself against any arguments which attack it.

Thus, the admissibility semantics can be understood in argumentation-theoretic terms. Given a theory T, an **argument** is simply a derivation in the underlying logic of some conclusion C from $T \cup N$, where N is some set of non-provability claims. In such a case, we also say both that N **supports the argument** and that the **argument is based on** N. One argument, based on N', is said to **attack** (or **undermine**) another argument, based on N, if, for some claim S <u>can not be shown</u> in N, the first argument derives the conclusion S. In such circumstances, we also say that N' is an (undermining) attack against N and that N' attacks (or undermines) the claim S <u>can not be shown</u>.

A candidate set N of non-provability sentences is said to be **admissible** if and only if
for every (undermining) attack N' against N, N attacks $N'-N$. [5]

Thus, an argument is admissible if and only if it is based upon an admissible set of non-provability claims.

Note that the stability semantics can also be understood in argumentation-theoretic terms, i.e. a set of non-provability claims is stable if and only if it does not attack itself and it attacks every non-provability claim it does not contain [2].

The definition of admissibility is a semantics in the sense that it specifies what is admissible, without indicating how such admissible sets of claims and arguments are to be constructed. For this latter purpose, we have developed an abstract proof procedure [23, 7], which applies to any instance of the framework.

The proof procedure is initiated by the proponent presenting an argument based upon sufficiently many non-provability claims N_0 to derive a desired conclusion C. The proponent then needs to defend N_0 against any attack N_0'

that an opponent might make. For this purpose, the proponent extends N_0 to a possibly larger set N_1 which, for every such attack N_0', attacks $N_0' - N_0$. The proponent then similarly needs to defend any new claims in $N_1 - N_0$ against all attacks. The process continues with the proponent generating an increasingly larger set of supporting claims $N_0, N_1, \ldots, N_n$.

The process terminates with a win for the proponent, with $N = N_n$, if N_n successfully counter-attacks all attacks by the opponent on the set of non-provability claims $N_n - N_{n-1}$. It terminates with a win for the opponent, if, for every sequence of moves $N_0, N_1, \ldots, N_n, \ldots$ by the proponent, there is an n for which the opponent has an attack against $N_n - N_{n-1}$ and the proponent has no counter-attack.

The formal definition of the proof procedure and a formal proof of its correctness and completeness, relative to the admissibility semantics, are given in [7].

6. Reduction of rebuttal attacks to undermining attacks

The abstract admissibility semantics and its dialectic proof procedure reduce all attacks to undermining attacks. In contrast, starting from [17], a number of argumentation-based frameworks [6, 18] allow, not only undermining attacks ("ground attacks" in [6]), but also

- **rebutting attacks** ("reductio-ad-absurdum attacks" in [6]), between arguments which derive contradictory conclusions.

Examples of such contradictory conclusions are *John is innocent of theft* and *John is* <u>not</u> *innocent of theft* in example 2.1 and *John inherits Henry's estate* and *John does* <u>not</u> *inherit Henry's estate* in example 2.2.

Undermining and rebutting attacks do not have equal status: undercutting attacks are stronger than rebutting attacks. Dung [6] motivates the difference between the status of the two kinds of attack by an example (example 1 in [6]) which has the form of the following variant of our example 2.2.

EXAMPLE 6.1. Consider the following two rules:

r_1: *A person inherits an estate*
<u>if</u> *he/she is the beneficiary of a valid will*
<u>and</u> <u>it can not be shown that</u> r_1 *is defeated for the person.*
r_2: *A person does* <u>not</u> *inherit an estate*
<u>if</u> *he/she has murdered the owner of the estate*
<u>and</u> <u>it can not be shown that</u> r_2 *is defeated for the person.*

the additional, third rule

r_1 *is defeated for a person*
<u>if</u> <u>it can not be shown that</u> r_2 *is defeated for that person.*

expresses the priority of the first rule (r_1) over the second (r_2). This rule might be justified, for example, by the general principle that no one should benefit from committing a crime. (In the next section, 7, we will show how to obtain this rule by compiling explicit priorities into non-provability claims.)

Suppose that, as in example 2.2, the facts

John is the beneficiary of a will.
John has murdered Henry.

are also given. Consider now the two conflicting arguments

Arg_1, based upon the set N_1 consisting of the single claim
<u>it can not be shown that</u> r_1 *is defeated for John*,
deriving the conclusion *John inherits an estate*, and

Arg_2, based upon the set N_2 consisting of the single claim
<u>it can not be shown that</u> r_2 *is defeated for John*,
deriving the conclusion *John does* <u>not</u> *inherit an estate*.

Arg_1/N_1 and Arg_2/N_2 attack each other via a rebutting attack. In addition, N_2 attacks Arg_1/N_1 via an undermining attack. If undermining attacks were not stronger than rebutting attacks, then the two alternative arguments would be equally admissible, and the priority given by the third rule would fail to be taken into account. Instead, if undermining attacks are stronger than rebutting attacks, then only Arg_2 is admissible.

In our argumentation framework only undercutting attacks are considered. In this example 6.1 we obtain the same intuitively correct result, as the more complicated frameworks, that only Arg_2 is admissible, since there is no undermining attack against Arg_2/N_2. On the other hand, Arg_1 is defeated, not by Arg_2, but by the third rule using the claims N_2 upon which Arg_2 is based.

The viability of our reduction of rebutting attacks to undermining attacks depends upon adopting a specific knowledge representation methodology, which introduces explicit non-provability conditions into rules. We will describe the full methodology, taking priorities into account, in the next section.

7. Reduction of priorities to non-provability conditions

Several authors, in particular Prakken and Sartor [18] and Dimopoulos and Kakas [4], have investigated extended logic programming with priority relations between rules. Other authors, such as Brewka [3], have studied the problem of assigning priorities to default rules more generally. Here, we present a methodology for dealing with priorities by adding extra non-provability conditions to rules, without changing the semantics. This methodology has been illustrated already, in condensed form, in examples 2.4 and 6.1.

For every rule which might be defeated by other rules we introduce a new predicate, which can be understood as naming the rule. The new predicate can also be interpreted as introducing a new concept standing for the way in which the rule establishes its conclusion.

For example, given several rules all of which imply that a particular person is a British citizen, the methodology introduces new predicates for each such rule. These new predicates can be thought of as representing new concepts, such as British citizenship by acquisition at birth, British citizenship by descent, or British citizenship by naturalisation, associated with different rules, all of which imply British citizenship. Arguably, the introduction of such predicates simply enables a more exact statement of the meaning of the originally given rules.

Thus each defeasible rule of the form

$$P(X) \ \underline{\text{if}} \ Q.$$

where X is a tuple of variables, is rewritten as two exact rules

$$\begin{array}{l} P(X) \ \underline{\text{if}} \ holds(rule(X)). \\ holds(rule(X)) \ \underline{\text{if}} \ Q \ \underline{\text{and}} \ defeated(rule(X)) \ \underline{\text{can not be shown}}. \end{array}$$

where $rule(X)$ can be understood both as the name of the original rule and as a more precise name for the conclusion of the rule.

Notice that if we do not need to establish separate conclusions of the form $holds(rule(X))$, then the two rules can be condensed into one

$$P(X) \ \underline{\text{if}} \ Q \ \underline{\text{and}} \ defeated(rule(X)) \ \underline{\text{can not be shown}}.$$

as in examples 2.4 and 6.1.

A rule is defeated if there is a conflicting rule which holds and has higher priority:

$$\begin{array}{l} defeated(R_1(X)) \\ \quad \underline{\text{if}} \ R_2(Y) > R_1(X) \\ \quad \underline{\text{and}} \ conflicting(R_1(X), R_2(Y)) \\ \quad \underline{\text{and}} \ holds(R_2(Y)). \end{array}$$

The priority relation can be defined by means of facts or by means of rules such as

$$r_1(X) > r_2(Y) \text{ \underline{if} } R.$$

Similarly, the predicate *conflicting* can be defined by means of facts or by means of rules of the form

$$\begin{aligned}&conflicting(R_1(X), R_2(X))\\ &\quad \underline{\text{if}}\ conclusion(R_1(X), P(X))\\ &\quad \underline{\text{and}}\ conclusion(R_2(X), \underline{\text{not}}\ P(X))\\ &conflicting(R_1(X,Y), R_2(X,Y))\\ &\quad \underline{\text{if}}\ conclusion(R_1(X,Y), X > Y)\\ &\quad \underline{\text{and}}\ conclusion(R_2(X,Y), Y > X).\end{aligned}$$

where $conclusion(R, P)$ holds when R is the rule P $\underline{\text{if}}$ Q.

Notice that, given two rules

$$\begin{aligned}&r_1(X)\text{: } P\ \underline{\text{if}}\ Q.\\ &r_2(X)\text{: } \underline{\text{not}}\ P\ \underline{\text{if}}\ R.\end{aligned}$$

with priority

$$r_2(X) > r_1(X)$$

the definition of *defeated* can be simplified to

$$defeated(r_1(X))\ \underline{\text{if}}\ R\ \underline{\text{and}}\ defeated(r_2(X))\ \underline{\text{can not be shown}}.$$

as in examples 2.4 and 6.1.

Our treatment of priority is illustrated by the following example, adapted from [18].

EXAMPLE 7.1.

Consider the following set of defeasible rules, including priority defining rules, before our transformation:

$r_1(X)$: *X's exterior may* <u>not</u> *be modified* <u>if</u> *X is a protected building.*
$r_2(X)$: *X's exterior may be modified* <u>if</u> *X needs restructuring.*
$r_3(X,Y)$: $R_1(X) > R_2(Y)$ <u>if</u> $R_1(X)$ *concerns artistic buildings*
<u>and</u> $R_2(Y)$ *concerns town planning.*
$T(X,Y)$: $R_1(X) > R_2(Y)$ <u>if</u> $R_1(X)$ *is later than* $R_2(Y)$.

and the following facts/exact rules:

$r_1(X)$ *concerns artistic buildings.*
$r_2(X)$ *concerns town planning.*
$r_2(X)$ *is later than* $r_1(X)$.
$r_3(X,Y)$ *is later than* $T(X,Y)$.

villa is a protected building.
villa needs restructuring.

After the transformation, and after simplifying the transformed rules by eliminating (using modus ponens) conditions that match the facts, we obtain

villa's exterior may not *be modified*
  if $holds(r_1(villa))$.
villa's exterior may be modified
  if $holds(r_2(villa))$.
$r_1(villa) > r_2(villa)$
  if $holds(r_3(villa, villa))$.
$r_2(villa) > r_1(villa)$
  if $holds(T(villa, villa))$.
$r_3(X,Y) > T(X,Y)$
  if $holds(T((X,Y),(X,Y)))$.

$holds(r_1(villa))$
  if $defeated(r_1(villa))$ can not be shown.
$holds(r_2(villa))$
  if $defeated(r_2(villa))$ can not be shown.
$holds(r_3(villa, villa))$
  if $defeated(r_3(villa, villa))$ can not be shown.
$holds(T(villa, villa))$
  if $defeated(T(villa, villa))$ can not be shown.
$holds(T((X,Y),(X,Y)))$
  if $defeated(T((X,Y),(X,Y)))$ can not be shown.

Using these rules, the definition of $defeated$ can be simplified first to

$defeated(r_1(villa))$
  if $holds(r_2(villa))$ and $holds(T(villa, villa))$.
$defeated(r_2(villa))$
  if $holds(r_1(villa))$ and $holds(r_3(villa, villa))$.
$defeated(T(villa, villa))$
  if $holds(r_3(villa, villa))$

<u>and</u> $holds(T((X,Y),(X,Y)))$.

After further simplification, eliminating the predicate *holds*, the original problem reduces to:

villa's exterior may <u>not</u> *be modified*
<u>if</u> $defeated(r_1(villa))$ <u>can not be shown</u>.
villa's exterior may be modified
<u>if</u> $defeated(r_2(villa))$ <u>can not be shown</u>.
$defeated(r_1(villa))$
<u>if</u> $defeated(T(villa, villa))$ <u>can not be shown</u>
<u>and</u> $defeated(r_2(villa))$ <u>can not be shown</u>.
$defeated(r_2(villa))$
<u>if</u> $defeated(r_3(villa, villa))$ <u>can not be shown</u>
<u>and</u> $defeated(r_1(villa))$ <u>can not be shown</u>.
$defeated(T(villa, villa))$
<u>if</u> $defeated(T((villa, villa), (villa, villa)))$ <u>can not be shown</u>
<u>and</u> $defeated(r_3(villa, villa))$ <u>can not be shown</u>.

We can now justify the conclusion that *villa's exterior may* <u>not</u> *be modified* by the following dialogue game.

Proponent: *villa's exterior may* <u>not</u> *be modified*
<u>because</u> $defeated(r_1(villa))$ <u>can not be shown</u>.

Opponent: On the contrary, $defeated(r_1(villa))$ can be shown
<u>because</u> $defeated(T(villa, villa))$ <u>can not be shown</u>
<u>and</u> <u>because</u> $defeated(r_2(villa))$ <u>can not be shown</u>.

Proponent: But, on the contrary, $defeated(r_2(villa))$ can be shown
<u>because</u> $defeated(r_3(villa, villa))$ <u>can not be shown</u>
<u>and</u> <u>because</u> $defeated(r_1(villa))$ <u>can not be shown</u>,
as I have already shown.

Alternatively, $defeated(T(villa, villa))$ can be shown
<u>because</u>
$defeated(T((villa, villa), (villa, villa)))$ <u>can not be shown</u>
<u>and</u> <u>because</u> $defeated(r_3(villa, villa))$ <u>can not be shown</u>.

Note that the proponent has two alternative ways to defeat the opponent. It is possible to show that the second argument is "more conclusive" than the first because it does not rely on its own claims to defeat the opponent. This argument can also be shown to be justified via a sceptical semantics as defined in [2].

The general methodology presented in this section applies also to the case of rules having contradictory conclusions where priorities are not given explicitly, as in example 2.2. In such cases, we treat each rule as having higher priority than every other rule with a contradictory conclusion. Applying the general transformation and simplifying the transformed rules in the manner of the discussion of this section, we obtain the same result as that illustrated in example 2.2.

For example, given the two defeasible rules

$P(X)$ if Q.
not $P(X)$ if R.

with contradictory conclusions, the general transformation introduces rule names, say $r_1(X)$ and $r_2(X)$, for the two rules and replaces the defeasible rules by four exact rules

$P(X)$ if $holds(r_1(X))$.
$holds(r_1(X))$ if Q and $defeated(r_1(X))$ can not be shown.
not $P(X)$ if $holds(r_2(X))$.
$holds(r_2(X))$ if R and $defeated(r_2(X))$ can not be shown.

Using the two priorities

$r_1(X) > r_2(X)$.
$r_2(X) > r_1(X)$.

the definition of $defeated$ simplifies to

$defeated(r_1(X))$ if $holds(r_2(X))$.
$defeated(r_2(X))$ if $holds(r_1(X))$.

Further simplifying, by removing some of the $holds$ conditions, we obtain

$P(X)$ if Q and $defeated(r_1(X))$ can not be shown.
$holds(r_1(X))$ if Q and $defeated(r_1(X))$ can not be shown.
not $P(X)$ if R and $defeated(r_2(X))$ can not be shown.
$holds(r_2(X))$ if R and $defeated(r_2(X))$ can not be shown.
$defeated(r_1(X))$ if $holds(r_2(X))$.
$defeated(r_2(X))$ if $holds(r_1(X))$.

But now the conclusions $holds(r_1(X))$ and $P(X)$ are interchangeable, as are $holds(r_2(X))$ and not $P(X)$. Therefore, rewriting $holds(r_1(X))$ as $P(X)$

and $holds(r_2(X))$ as not $P(X)$, we obtain

$P(X)$ if Q and $defeated(r_1(X))$ can not be shown.
not $P(X)$ if R and $defeated(r_2(X))$ can not be shown.
$defeated(r_1(X))$ if not $P(X)$.
$defeated(r_2(X))$ if $P(X)$.

Eliminating the predicate $defeated$ gives

$P(X)$ if Q and not $P(X)$ can not be shown.
not $P(X)$ if R and $P(X)$ can not be shown.

which is exactly the result of the simplified transformation needed to convert defeasible rules into exact rules, when no priorities are explicitly given. This simplification can be generalised to the general case where there are several rules having conclusion $P(X)$ or not $P(X)$.

In all the examples considered up until now, we obtain the same results as those obtained by Prakken and Sartor [18] (but we employ a credulous semantics while they employ a sceptical one). In general, however, our treatment of priorities gives different results from those of Prakken and Sartor [18], as illustrated by the following example.

EXAMPLE 7.2. (Taken from [18])
Consider the defeasible rules

r_1: A.
r_2: B if A.
r_3: not A if B can not be shown.

with $r_3 > r_1$. Then, according to [18], the derivation consisting of r_1 and r_2 is a justified argument, while the derivation consisting of r_3 and the claim B can not be shown is not justified. Therefore, they derive A but not not A. Instead, our method produces three admissible arguments:

Arg_1 concluding A,
based upon the set of claims N_1 consisting of
$defeated(r_1)$ can not be shown and $defeated(r_2)$ can not be shown,

Arg_2 concluding not A,
based upon the set of claims N_2 consisting of
B can not be shown and $defeated(r_3)$ can not be shown,

Arg_3 concluding not A,

based upon the set of claims N_3 consisting of
B can not be shown, $defeated(r_3)$ can not be shown and
$defeated(r_2)$ can not be shown.

Note that our approach takes into account the given priority, while their approach does not.

8. Conclusion

We have argued that our abstract framework for defeasible reasoning, with the admissibility semantics and its dialectic proof procedure, is adequate for the formalisation of many aspects of practical reasoning. We have outlined a methodology for transforming inexact, defeasible rules into exact rules with explicit non-provability conditions; and we have argued that this transformation eliminates the need for rebuttal attacks and for dealing with priorities in the semantics. Our transformation is an elaboration of transformations [14, 24] we have developed earlier.

Because our approach is abstract, it can be formalised in any one of the many formalisms, including default logic, extended logic programming, non-monotonic modal logic, and auto-epistemic logic, which are special cases of our framework.

It needs to be noted, however, that our representation of priorities uses meta-predicates, such as "holds" and "defeated", in the manner of meta-logic programming [1]. In this respect, our approach bears many resemblances to that of Hage [11], who uses similar meta-predicates. However, his approach differs form ours in his introduction of a new formalism for this purpose, while our approach can be employed with many existing formalisms. Moreover, as we saw in several examples and in one general case, in our simple use of them, these meta-predicates can generally be eliminated in favour of simpler non-provability conditions.

In this paper, we focussed entirely on the credulous admissibility semantics. However, as has been shown elsewhere [2], the abstract framework also admits sceptical semantics and proof procedures.

Although we have given several examples to show how our methodology eliminates the need for rebuttal attacks and for dealing with priorities in the semantics, we do not have any formal results proving that the methodology is always adequate. This remains to be done for future work.

Acknowledgements

This research was partially supported by Fujitsu Research Laboratories. The authors are grateful to Phan Minh Dung and Henry Prakken for many helpful discussions, and to the referees and Henry Prakken for helpful suggestions.

Notes

[1] We use the phrases "S can not be shown" and "it can not be shown that S interchangebly".

[2] As we will see later, non-monotonic modal logic and autoepistemic logic are exceptions.

[3] In non-monotonic modal logic and autoepistemic logic, non-provability sentences not in N may be derivable from $T \cup N$. In such cases, for N to be stable, N must be **closed** in the sense that it contains all non-provability sentences derivable from $T \cup N$.

[4] In conventional presentations of modal logics, if is written as $\subset$.

[5] In nonmonotonic modal logic and some other logics, where non-provability sentences not in N may be derivable from $T \cup N$, the sets N and N' in the definition of admissibility should be closed (see footnote 3).

References

1. *Meta-logics and logic programming.* MIT Press (1995) (K. Apt and F. Turini eds).
2. A. Bondarenko, P. M. Dung, R. A. Kowalski, F. Toni, An abstract, argumentation-theoretic framework for default reasoning. To appear in *Artificial Intelligence*, Elsevier.
3. G. Brewka, Preferred subtheories: an extended logical framework for default reasoning. *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, Mi (1989) Morgan Kaufmann (N. Sridharan, ed.) 1043–1048
4. Y. Dimopoulos, A.C. Kakas, Logic programming without negation as failure. *Proceedings of the International Logic Programming Symposium*, Portland, Oregon (1995) MIT Press (J. Lloyd, ed.) 369–384
5. P.M. Dung, The acceptability of arguments and its fundamental role in non-monotonic reasoning and logic programming. *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambery, France (1993) Morgan Kaufmann (R. Bajcsy, ed.) 852–857
6. Dung, P.M., An argumentation semantics for logic programming with explicit negation. *Proceedings of the 10th International Conference on Logic Programming*, Paris (1993) MIT Press (K. Furukawa, ed.)
7. P.M. Dung, R.A. Kowalski, F. Toni, Argumentation-theoretic proof procedures for non- monotonic reasoning. *Proc. 6th LOgic Programming Synthesis and TRansformation* (1996) (J. Gallagher ed.)
8. M. Gelfond, V. Lifschitz, The stable model semantics for logic programming. *Proceedings of the 5th International Conference on Logic Programming*, Washington, Seattle (1988) MIT Press (K. Bowen and R.A. Kowalski, eds.) 1070–1080
9. M. Gelfond, V. Lifschitz, Logic programs with classical negation. *Proceedings of the 7th International Conference on Logic Programming*, Jerusalem (1990) MIT Press (D.H.D. Warren and P. Szeredi, eds.) 579–597
10. T.F. Gordon, The pleadings game: an exercise in computational dialectics. *Artificial Intelligence and Law* 2(4), Kluwer Academic Publishers (1993-1994) 239–292
11. J. Hage, Teleological reasoning in reason-based logic. *Proceedings of the 5th International Conference on Artificial Intelligence and Law*, College Park, MD (1995) ACM Press, 11–20
12. K. Konolige, Autoepistemic logic. *Handbook of Logic in Artificial Intelligence and Logic Programming* 3, Oxford University Press (1994) (D. Gabbay, C. Hogger, J.A. Robinson, eds.)

13. R.A. Kowalski, Legislation as logic programs. *Informatics and the Foundations of Legal Reasoning*, Kluwer Academic Publishers (1995) (Z. Bankowski et al., eds.) 325–356
14. Kowalski, R.A., Sadri, F., Logic programs with exceptions. *Proceedings of the 7th International Conference on Logic Programming*, Jerusalem (1990) MIT Press (D.H.D. Warren and P. Szeredi, eds.) 598–613
15. D. McDermott, Nonmonotonic logic II: non-monotonic modal theories. *Journal of ACM* 29(1) (1982) 33–57
16. R. Moore, Semantical considerations on non-monotonic logic. *Artificial Intelligence* 25, Elsevier (1985) 75–94
17. J.L. Pollock, Defeasible reasoning. *Cognitive Science*, 11 (1987) 481–518
18. H. Prakken, G. Sartor, On the relation between legal language and legal argument: assumptions, applicability and dynamic priorities. *Proceedings of the 5th International Conference on Artificial Intelligence and Law* College Park, MD (1995) ACM Press, 1–10
19. H. Prakken, G. Sartor, A dialectical model of assessing conflicting arguments in legal reasoning. In this issue.
20. R. Reiter, A logic for default reasoning. *Artificial Intelligence* 13, Elsevier (1980) 81–132
21. G. Sartor, The structure of norm conditions and non-monotonic reasoning in law. *Proceedings of the 3th International Conference on Artificial Intelligence and Law*, Oxford (1991) ACM Press, 155–164
22. G. Shvarts, Autoepistemic modal logics. *Proc. 3rd Conference on Theoretical Aspects of Rationality and Knowledge*, Pacific Grove, CA (1990) Morgan Kaufmann (R. Parikh, ed.) 97–110
23. F. Toni, A.C. Kakas, Computing the acceptability semantics. *Proceedings of the 3rd International Workshop on Logic Programming and Non-monotonic Reasoning,* Springer Verlag LNAI 928 (1995) (V. Marek, A. Nerode, M. Truszczynski, eds.) 401–415
24. F. Toni, R.A. Kowalski, Reduction of abductive logic programs to normal logic programs. *Proceedings of the 12th International Conference on Logic Programming*, Japan (1995) MIT Press (Leon Sterling, ed.) 367–381