A Dialectical Model of Assessing Conflicting Arguments in Legal Reasoning

H. Prakken *
Computer/Law Institute, Faculty of Law, Free University Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands. email: henry@rechten.vu.nl

G. Sartor
Faculty of Law, The Queen's University of Belfast, Belfast BT7 1NN, Northern Ireland. email: sartor@cirfid.unibo.it

July 19, 1996

Abstract. Inspired by legal reasoning, this paper presents a formal framework for assessing conflicting arguments. Its use is illustrated with applications to realistic legal examples, and the potential for implementation is discussed. The framework has the form of a logical system for defeasible argumentation. Its language, which is of a logic–programming–like nature, has both weak and explicit negation, and conflicts between arguments are decided with the help of priorities on the rules. An important feature of the system is that these priorities are not fixed, but are themselves defeasibly derived as conclusions within the system. Thus debates on the choice between conflicting arguments can also be modelled.

The proof theory of the system is stated in dialectical style, where a proof takes the form of a dialogue between a proponent and an opponent of an argument. An argument is shown to be justified if the proponent can make the opponent run out of moves in whatever way the opponent attacks. Despite this dialectical form, the system reflects a 'declarative', or 'relational' approach to modelling legal argument. A basic assumption of this paper is that this approach complements two other lines of research in AI and Law, investigations of precedent–based reasoning and the development of 'procedural', or 'dialectical' models of legal argument.

Key words: Argumentation, defeasibility, dialectics, rule conflicts, logic programming

1. Introduction

In recent years, adversarial legal argumentation has been a major topic of AI–and–Law research, approached from different angles. Some researchers, e.g. (Ashley, 1990; Rissland & Skalak, 1991; Skalak & Rissland, 1992), have developed theories or programs that are mainly descriptive, in that they are meant to model the way lawyers actually argue. Others, including the present authors, have studied the logical notions involved in legal argumentation, and again others, e.g. (Hage et al., 1994; Gordon, 1994) have formalised the role of legal procedures

* Supported by a research fellowship of the Royal Netherlands Academy of Arts and Sciences, and by Esprit WG 8319 'Modelage'.

in legal argument[1]. In our view, these approaches complement each other: descriptive theories and programs need logical and procedural standards for correct argumentation, and procedural models of dispute are built around a logical theory of notions like 'argument', 'counterargument' and 'defeat'. Finally, logical and procedural theories of argumentation are sterile if they are not susceptible of being embedded in actual reasoning practice.

The present paper extends the logical line of research. Our aim is to develop a formal theory on the assessment of conflicting arguments that takes as many argumentation features of legal language into account as possible. This is based on our view that strategies of attack and defence in dispute are partly based on the logical aspects of the language in which arguments are stated. In particular, lawyers can take advantage of some specific features of legal language, such as 'unless' clauses, applicability statements and statements about preference relations between legal rules.

This paper is also relevant for theories about non–adversarial aspects of legal reasoning. The structures we study are often deliberately used by the legislator as a technique for drafting well–structured legal texts or as a way of anticipating changes in the law. If the aim of a computational representation is to maintain these features, then our work points at ways of doing so.

While introducing complexity in the argumentative features of our language, we reduce complexity in another respect: syntactically, our system will be restricted in logic–programming style. In particular, we will build on (Dung, 1993)'s argumentation–theoretic semantics of extended logic programming. We make this restriction in order to make the discussion of the other types of complexity more transparent, but also to reduce the computational complexity and increase the potential for implementation.

The theory presented in this paper, which was presented in an earlier form in (Prakken & Sartor, 1995) and with more technical details in (Prakken & Sartor, 1996a), extends our own previous work, reported in e.g. (Prakken, 1991; Prakken, 1993; Sartor, 1993; Sartor, 1994). (Prakken & Sartor, 1995; Prakken & Sartor, 1996a) contained two main additions to this work. Firstly, a weak negation operator was used to model the fact that legal reasoning combines the use of priorities to choose between conflicting rules, with the use of 'unless' or similar phrases within a rule to make it inapplicable in exceptional circumstances. Secondly, the means to reason *about* priorities was introduced, i.e. the possibility to derive the ordering on the rules from the rules themselves. This captures the fact that in the legal domain everything is debatable, including the standards for comparing arguments.

This article adds two main features to (Prakken & Sartor, 1995; Prakken & Sartor, 1996a). It adds the category of strict rules to the language, and it replaces the earlier fixpoint definition of a justified argument with a constructive proof theory in dialectical style, which is inspired by work of (Vreeswijk, 1993b; Dung, 1994; Brewka, 1994b; Loui & Norman, 1995). The proof theory is defined as a dialogue game: a proof takes the form of a dialogue between a proponent and an opponent of an argument, where each player attempts to defeat the previous move of the other player; an argument is shown to be justified if the proponent can make the opponent run out of moves in whatever way the opponent attacks. A companion paper of the present article is (Prakken & Sartor, 1996b), which has less emphasis on legal issues and examples but contains more formal details and results.

The structure of this paper is as follows. We will start in the next section by putting our work in the context of related argument–based developments in AI and Law. Then, in section 3, we will discuss some structural features of legal language that we want our system to be able to deal with. After that, the system is presented and applied to a wide range of examples in sections 4 to 7. These sections form the heart of the paper. After some remarks on implementation in section 8, we end with a discussion of related research and some concluding remarks.

2. Argumentation frameworks: general ideas and their role in models of legal argument

Our system takes the form of a system for defeasible argumentation, or an 'Argumentation Framework' (AF). In this section we discuss the general ideas behind AFs, and we also discuss how research on AFs relates to other argument–oriented developments in AI and Law research.

Most AFs have been developed in general Artificial–Intelligence research on nonmonotonic reasoning, although Pollock's work on defeasible argumentation, e.g. in (Pollock, 1992), was developed to analyze epistemological issues in the philosophy of science. In AI argument–based systems have been developed as a reformulation of, (Bondarenko et al., 1993; Dung, 1995), or an alternative to, (Loui, 1987; Simari & Loui, 1992; Vreeswijk, 1993a), earlier formalisations of nonmonotonic reasoning. The idea here is that nonmonotonic reasoning can be analyzed in terms of the interactions between arguments for alternative conclusions. Nonmonotonicity, or defeasibility, arises from the fact that arguments can be defeated by stronger counterarguments. As is commonly accepted in AI and Law, legal reasoning is also defeasible, and since in our domain notions like argument, counterargument, rebuttal and defeat are very common, it comes as no surprise that several AFs have already been applied to the legal domain (Prakken, 1991; Prakken, 1993; Loui et al., 1993; Sartor, 1993; Gordon, 1994; Gordon, 1995; Prakken & Sartor, 1995).

To describe the general structure of AFs, they contain the following five elements, although sometimes implicitly: an underlying logical language, definitions of an argument, of conflicts between arguments and of defeat among arguments and, finally, an assessment of arguments.

AFs are built around an underlying logical language. Some AFs, like ours, assume a particular logic, while other systems, e.g. (Vreeswijk, 1993a; Bondarenko et al., 1993; Dung, 1995), leave the underlying logic partly or wholly unspecified; thus these systems can be instantiated with various alternative logics, as (Dung, 1993) has done for extended logic programming. Then AFs have the notion of an argument, which corresponds to a proof in the underlying logic. This is a narrow use of the term 'argument', which should not be confused with the broader meaning it often has in AI and law, when it also includes inductive, abductive and especially analogical arguments.

The notions of an underlying logic and an argument still fit with the standard picture of what a logical system is. The remaining three elements are what makes an AF a framework for adversarial argumentation. The first is the notion of a conflict between arguments. The typical case is when arguments have contradictory conclusions. An AF also has ways of comparing arguments, on the basis of certain standards, to see whether an argument defeats a counterargument. In AI the specificity principle is regarded as very important, but one of the main themes of this paper is that any standard might be used: their content is part of the domain theory, and is debatable, just as the rest of the domain theory is.

Since attacking arguments can themselves be attacked by other arguments, comparing pairs of arguments is not sufficient; what is also needed is a definition of the status of arguments on the basis of all ways in which they interact. It is this definition that produces the output of an AF: it divides arguments in, typically, three classes: arguments with which a dispute can be 'won', respectively, 'lost' and arguments which leave the dispute undecided. In this paper we will denote these classes with the terms 'justified', 'overruled' and 'defensible' arguments.

Systems which fit this description have been called 'declarative' (Loui & Norman, 1995) and 'relational' (Gordon, 1994; Gordon, 1995). The reason is that AFs are not intended to model the *process* of argumentation; i.e. they are not meant to define standards for fair and effective dispute, or to define strategies for adding new information into the debate. Instead AFs define the nature of the relation between the premises put forward at a certain stage of a dispute, and the possible conclusions at that stage; i.e. they declare an argument to be in one of the three classes on the basis of the premises and the standards (for comparing arguments) that are given at a certain stage of a dispute.

Two other developments in AI and Law have focused more on the process of argumentation, in the setting of a two–party adversarial dialogue. The first development is research in precedent–based reasoning of e.g. (McCarty & Sridharan, 1982; Ashley, 1990; Rissland & Skalak, 1991; Skalak & Rissland, 1992). where the dialogue moves are possible dialectical uses of precedents, such as analogizing, distinguishing and rule discrediting. Whereas this line of research mainly focuses on the specific moves that can be made with precedents, and on the ways in which the precedents can best be represented to support these moves, another development, e.g. (Gordon, 1994; Gordon, 1995; Lodder & Herczog, 1995), focuses more on the general procedural properties of disputes. The leading idea here is that justice and rationality have a procedural side: a legal argument is acceptable if it has been defended against criticism in a properly conducted dispute. The main aims of this line of research are to define the general form of dialectical protocols, and to study what makes such a protocol proper, i.e. what makes it fair and effective.

As already mentioned in the introduction, and as earlier defended in (Prakken, 1995b), we think that these three developments complement each other. A dialectical model of fair and effective legal argument needs an argumentation framework to determine at each stage of a dispute which party is winning, and logical and procedural models are sterile if not complemented with theories on generating particular disputes in a realistic way. On the other hand, the logical and procedural models formulate standards with which actual disputes should comply.

An interesting attempt to synthesize these approaches is (Loui & Norman, 1995), which defines the allowed types of moves in a dispute, uses the AF of (Simari & Loui, 1992) to assess arguments at each stage of the debate, and presents ways of exploiting rationales of precedents to introduce new or remove old pieces of input information.

Finally, note that from this description two notions of an argument have emerged: the logical notion of an argument as a necessary relation between premises and a conclusion, and the dynamic notion of an argument as a move in a disputational setting. We think that these notions complement each other; in particular, we think that analogical arguments, as studied in e.g. (Ashley, 1990), can best be conceived as types of such argument moves, i.e. as heuristics for adding premises (for more detail see (Prakken, 1995b)).

3. Attacking points in legal language

We now illustrate with some examples the features of legal language that we aim to formalise. Legal language offers a wide inventory of argumentative structures, such as contradictory rules, rules with assumptions, inapplicability statements, and priority rules. These structures can be found in legal discussions and documents recording them, but also in non argumentative legal documents, such as statutory texts or contracts.

Sometimes different arguments use rules with contradictory consequents. In statutory texts such 'head–to–head' conflicts can be found for a variety of reasons, extensively discussed in our earlier work. In legal debates they typically arise when parties appeal to competing principles, as in the following example from Italian law. According to the principle of the protection of privacy, it is forbidden to propagate private information, while according to the principle of the freedom of communication, it is allowed to propagate every piece of information, at least when it has a public significance. A piece of information about the private life of a person having a public role both concerns the privacy of an individual and has public relevance, so that its propagation puts the two just mentioned principles into conflict.

Another example of this type concerns a conflict between European and Italian law. According to a general principle of the European community law, if a product is lawfully put into commerce in an E.C. country, it can be freely sold in any other E.C. country, unless special reasons occur, such as the preservation of public health or consumer protection. In Italy it had been forbidden to sell products called 'pasta' not being made of hard corn. According to a decision of the European court of justice European law prevails, and 'pasta' products not made of hard corn, merchandized in other European countries, can also be sold in Italy.

The next feature we want to formalise is reasoning about priorities. To decide a head–to–head conflict between arguments with incompatible rules often priority rules are used, establishing which one of the competing rules is to prevail. The law offers well–known general priority rules such as the principles of specificity (Lex Specialis), temporality (Lex Posterior) and hierarchy (Lex Superior). Other domain–dependent general priority rules are explicitly stated by the legislator. For instance, in Italy, in town planning regulations we can find a priority rule stating that rules intended to protect artistic buildings prevail over rules concerning town planning; and in The Netherlands section 1637c of the Civil Code gives priority to statutory rules concerning labour contracts over statutory rules concerning any other type of contract.

Besides general priority rules, there are implicit specific priority assertions, which the interpreter may detect, just by a combined interpretation of the involved rules. For example, the Italian Criminal Code includes a prescription establishing that he who acted in self defence is not responsible. This prescription is obviously (although implicitly) intended to prevail over each particular criminal provision establishing responsibility for a specific crime.

Finally, substantive priority rules may be used, adducing special domain dependent grounds for preferring certain rules. For instance, in the privacy example the conflict could be solved by a rule stating that if the involved person is a politician, and the private information concerns aspects which may affect his/her political functioning, then the communication rule should be preferred.

It is important to note that priority rules behave exactly like any other legal rule: also these rules make their consequent (a priority assertion) dependent on the satisfaction of certain conditions, and also between these rules the conflicts can be solved appealing to priority rules. For instance, the town planning priority rule may be in conflict with Lex Posterior and to this conflict Lex Superior, Lex Specialis and, as we will see in section 6.2, even Lex Posterior itself may apply. Our formalisation will respect this, i.e. it will treat priority rules like any other premise.

We will also include assumptions in our formalisation. Often, the satisfaction of the antecedent provided by a legal rule is able to produce the effect of the rule only if certain conditions are not proved to hold: the complements of these conditions will here be called the assumptions of the rule. For instance, in the above pasta example the EC rule contains a clause 'unless special reasons occur'. Another example is section 3:32–(1) of the Dutch Civil Code, which declares every person to have the capacity to perform juridical acts, "to the extent that the law does not provide otherwise". Often the methods of representing exceptions explicitly, with assumptions, and implicitly, with priorities, are presented as alternatives. However, in legal texts also rules with explicit exceptions are subject to defeat by rules with higher priority, for instance, on the basis of Lex Superior or Lex Posterior; therefore a full theory of legal argument should be able to deal with the combination of assumptions and priorities.

In some cases assumptions may be 'discovered' by the legal interpreter; then they are considered implicit presuppositions of the involved rules. Other assumptions are explicitly established by the involved rule, by including an unless clause or a similar expression, as, as in the two above–given examples.

Finally, we want to formalise applicability rules. In legal texts and judicial debates rules frequently occur which deny the applicability of certain other rules, under certain conditions. Those rules do not exclude a determinate legal conclusion, but intend just to exclude the possibility of using a specific rule or set of rules. For instance, section 2 of the Dutch rent act states that the act is not applicable to lease contracts which by their nature concern a short termed usage. Note that rules denying applicability may themselves be contradicted by rules affirming it.

4. The formal system I: fixed priorities

We now present our own argumentation framework, along the lines of the informal description in section 2. We will present the system in two stages: in this and the following section we will assume that the priorities are fixed and undisputable, while in section 6 we will make them defeasible, by allowing also arguments about the priorities.

4.1. The language

The object language of our system is of familiar logic- programming style: it contains a twoplace one–direction connective that forms rules out of literals, which can contain two kinds of negation, weak and strong negation. A *strong* literal is an atomic first-order formula, or such a formula preceded by strong negation $\neg$. For any atom A we say that A and $\neg A$ are the complement of each other; in the metalanguage we will denote the complement of a literal L with $\overline{L}$. A *weak* literal is a literal of the form $\sim L$, where L is a strong literal. Informally, $\sim L$ reads as 'there is no evidence that L is the case', while $\neg L$ says 'L is definitely not the case'.

DEFINITION 4.1. A *rule* is an expression of the form

$$r : L_0 \wedge \ldots \wedge L_j \wedge \sim L_k \wedge \ldots \wedge \sim L_m \Rightarrow L_n$$

where r, a first–order term, is the name of the rule and each L_i ($0 \leq i \leq n$) is a strong literal. The conjunction at the left of the arrow is the *antecedent* and the literal at the right of the arrow is the *consequent* of the rule. As usual, a rule with variables is a scheme standing for all its ground instances.

Note that a rule may have zero literals in its antecedent. Such a rule can be used to express an unconditional statement. Although informal discourse might prefer to call unconditional rules 'facts' (when they are strict) or 'assumptions' (when they are defeasible), we will also call such rules 'rules', to stress that in our system they are treated in exactly the same way as proper conditional rules.

Variables will in this paper be denoted with the letters x, y and z, and constants with any other letter, or with words.

Next we define the input information of our system, which will be called an *ordered theory*. Obviously, the input contains rules. However, we will assume that the input rules are divided into two categories, the set D of *defeasible* rules and the set S of *strict* rules; we assume that no member of S contains a weak literal. Informally, the defeasible rules express information that is intended by the user of the system to be subject to debate, while the strict rules represent the information that is intended to be beyond debate. Thus the strict rules can be used to express meaning postulates, like 'Bachelors are not married', and properties of relations, like transitivity of the 'older than' relation or asymmetry of the 'father of' relation. Note that a defeasible rule can be defeasible in two ways: it can be overridden by stronger rules with a contradicting consequent, and it can contain assumptions that are not warranted.

Syntactically, strict and defeasible rules differ in only one respect: only defeasible rules can contain assumptions. For the rest we have chosen to make the distinction between strict and defeasible rules not at the syntactic but at the pragmatic level. That is, the logic governing the conditional operator is the same for both types of rules; they differ only in their epistemological status. Yet, for notational convenience we will, when writing down rules, use $\rightarrow$ when a rule is strict and $\Rightarrow$ when a rule is defeasible. However, formally the symbol $\rightarrow$ is not part of our object language; it is just a shorthand for saying that a rule is in the set of strict rules.

As discussed in the introduction, in legal reasoning conflicts between defeasible rules are often resolved with the help of statements on which rule takes precedence. Accordingly, our input information will also contain an ordering $<$ on the set D, which we will assume to be a strict partial order, i.e. transitive (if $x < y$ and $y < z$ then $x < z$) and asymmetric (if $x < y$ then $y \not< x$). $x < y$ means that y has priority over x. Because strict rules are beyond debate, no ordering needs to be defined on S.

In sum, our system assumes input information in the form of an *ordered theory* $(S, D, <)$, where S and D are sets of, respectively, strict and defeasible rules and $<$ is a strict partial order on D.

4.2. Arguments

The basic notion of a system for defeasible argumentation is that of an argument. In general, the idea is that an argument for a certain proposition is a proof of that proposition in the logic of the underlying language. In our system, the simple language gives rise to a simple notion of an argument, viz. as a sequence of rules that can be chained together, and that is 'grounded' in the facts. This is captured by a slight variant of (Dung, 1993)'s notion of a 'defeasible proof'.

DEFINITION 4.2. An *argument* is a finite sequence $A = [r_0, \ldots, r_n]$ of ground instances of rules such that

1. for every i $(0 \leq i \leq n)$, for every strong literal L_j in the antecedent of r_i there is a $k < i$ such that L_j is the consequent of r_k.
2. No two distinct rules in the sequence have the same consequent.

An argument A is *based on* the ordered theory $(S, D, <)$ iff all rules of A are in $S \cup D$.[2]

Condition (1) says that arguments are formed by chaining rules together. A subtlety is that in doing so, weak literals may be ignored. Condition (2) prevents arguments from containing circular chains of rules.

For any ordered theory Γ we will denote the set of all arguments on the basis of Γ with $Args_\Gamma$. Likewise, for any set R of rules, $Args_R$ stands for the set of all arguments that consist of only rules in R.

We will also use the following notions.

DEFINITION 4.3. For any argument A:

1. A is *strict* iff it does not contain any defeasible rule; it is *defeasible* otherwise.
2. An argument A' is a *(proper) subargument* of A iff A' is a (proper) subsequence of A.
3. a literal L is a *conclusion* of A iff L is the consequent of some rule in A;
4. A literal L is an *assumption* of A iff $\sim \overline{L}$ occurs in some rule in A.

Note that Condition (3) makes the consequent of any rule in the chain a conclusion of the argument. The alternative, to regard only the consequent of the last rule as a conclusion, is perhaps philosophically more usual, but our choice makes some other definitions simpler.

To illustrate these notions, for the (defeasible) argument

$$A = [r_1\colon \rightarrow a,\ r_2\colon a \wedge \sim \neg b \Rightarrow c,\ r_3\colon c \wedge \sim d \Rightarrow e]$$

we have that A's conclusions are $\{a, c, e\}$, its subarguments are $\{[\ ], [r_1], [r_1, r_2], [r_1, r_2, r_3]\}$, of which only $[\ ]$ and $[r_1]$ are strict, and A's assumptions are $\{b, \neg d\}$.

EXAMPLE 4.4. The next example illustrates the distinction between weak and strong negation. Rule r_1 states that a person who cannot be shown to be a minor has the capacity to perform legal acts, and rule r_2 requires that a person is positively shown not to be a minor, in order that s/he can exercise the right to vote. The point is that these rules give rise to an argument for 'x has legal capacity' but not for 'x has the right to vote', since there is no rule providing the antecedent of r_2.

r_1: $\sim x$ is a minor $\Rightarrow x$ has legal capacity
r_2: $\neg\ x$ is a minor $\Rightarrow x$ has the right to vote

4.3. Conflicts between arguments

So far the notions have been fairly standard; now we will present the adversarial aspects of our system. We first define in which ways an argument can attack, i.e. be a counterargument of another argument. This definition does not yet include any way of checking which argument is better; it only tells us which arguments are in conflict. As already indicated in section 3, the two different kinds of negation give rise to two ways of attacking an argument. Let us first explain them informally. Assume an argument A has a conclusion L. Then the first way of attack consists in contradicting a conclusion: i.e. A attacks any argument B which has a conclusion $\overline{L}$. Clearly, such a head–to–head conflict is symmetric: if A head–to–head attacks B, then B head–to–head attacks A. The second way of attack, with weak negation, consists in contradicting an assumption: i.e. A attacks any argument B which has an assumption $\overline{L}$. Obviously, this way of attack is not symmetric.

However, strict rules complicate the matter. To see this, consider the following example.

EXAMPLE 4.5. Assume we have two defeasible rules

$r_1 : A \Rightarrow x$ is a lease
$r_2 : B \Rightarrow \neg\ x$ is a contract

and a strict rule

s_1 : x is a lease $\rightarrow$ x is a contract

and assume we have the strict facts f_1: $\rightarrow A$, f_2: $\rightarrow B$. Now the rules with complementary consequents are[3] r_2 and s_1. However, intuitively it seems that r_2 does not compete with the strict rule s_1 but with the defeasible rule r_1; s_1 just expresses a linguistic convention.

Here is a variant of this example.

EXAMPLE 4.6. Assume we have

r_1 : $A \Rightarrow$ x is married
r_2 : $B \Rightarrow$ x is a bachelor
s_1 : x is married $\rightarrow \neg$ x is a bachelor
s_2 : x is a bachelor $\rightarrow \neg$ x is married

and the strict facts f_1: $\rightarrow A$, f_2: $\rightarrow B$. Again, the rules that intuitively are in conflict with each other are not the rules with complementary consequents (r_1 v. s_2 and r_2 vs. s_1), but the two defeasible rules r_1 and r_2. The strict rules just declare the predicates 'married' and 'bachelor' to be incompatible.

These examples suggest the following way of defining a conflict between arguments: if an argument A has a conclusion or assumption L, we should consider A as being attacked by all those arguments which can *with only strict rules* be extended to an argument with conclusion $\overline{L}$.

To illustrate this, in Example 4.5 the argument $[f_1, r_1]$ attacks the argument $[f_2, r_2]$, since $[f_1, r_1]$ can with the strict rule s_2 be extended to an argument for 'x is a contract'. More generally, in the first way of attack we have to extend *both* arguments with only strict rules, to see if eventually they will have complementary conclusions. This is captured by the following definitions, which, like all other definitions, are implicitly assumed to be relative to an ordered theory $(S, D, <)$. First we have to define the notion of extending an argument.

DEFINITION 4.7. Let A be an argument and T a sequence of rules. Then $A + T$ is the concatenation of A and T.[4]

Now the notion of attacking/conflicting/counterarguments is defined as follows.[5]

DEFINITION 4.8. Let A_1 and A_2 be two arguments. Then A_1 *attacks* A_2 iff there are sequences S_1, S_2 of strict rules such that $A_1 + S_1$ is an argument with conclusion L and

1. $A_2 + S_2$ is an argument with a conclusion $\overline{L}$; or
2. A_2 is an argument with an assumption $\overline{L}$.

We have already discussed some peculiarities of this definition when strict rules are involved. We can add that if two arguments contain defeasible rules with complementary consequents, the definition is satisfied if S_1 and S_2 are empty. We now give some examples of this type, illustrating some more features of Definition 4.8.

EXAMPLE 4.9. The first example shows that in order to attack an argument, a counterargument can point its attack at that argument itself, but also at one of its proper subarguments, thereby indirectly attacking the entire argument. So if we have

r_1: $\Rightarrow f$ forged evidence e
r_2: f forged evidence $e \Rightarrow \neg\ e$ is admissible evidence
r_3: $\Rightarrow \neg\ f$ forged evidence e

we have that $[r_3]$ does not only attack $[r_1]$, but also $[r_1, r_2]$.

EXAMPLE 4.10. And if we vary this example into

r_1: $\Rightarrow f$ forged evidence e
r_2: f forged evidence $e \Rightarrow \neg\ f$ is honest
r_3: f is police officer $\Rightarrow f$ is honest
r_4: f is honest $\Rightarrow \neg\ f$ forged evidence

together with the fact

s_1: $\rightarrow f$ is police officer

then we have that $[r_1, r_2]$ does not only attack $[s_1, r_3]$ but also $[s_1, r_3, r_4]$. And $[s_1, r_3, r_4]$ does not only attack $[r_1]$, but also $[r_1, r_2]$.

The concept of 'attack/counterargument' is very important, since clearly a minimum requirement on any system for defeasible argumentation is that if two arguments are in conflict with each other, they cannot both be accepted as justified: at best, they can be accepted alternatively. We want our system to agree with this, in the sense that the set of 'justified arguments', to be defined later, is unique and conflict–free. To this end, we now define the following notions.

DEFINITION 4.11. An argument is *coherent* iff it does not attack itself.

Two examples of incoherent arguments are $[r_1: \Rightarrow a,\ r_2: a \Rightarrow \neg a]$ and $[r_1: \sim a \Rightarrow b,\ r_2: b \Rightarrow a]$.

DEFINITION 4.12. A set $Args$ of arguments is *conflict–free* iff no argument in $Args$ attacks an argument in $Args$.

COROLLARY 4.13. *If $Args_S$ is not conflict–free, then every argument is incoherent.*

4.4. Defeat among arguments

Now that we know which arguments are in conflict with each other, the next step is to compare conflicting arguments. To this end we define, in two steps, a binary relation of defeat among arguments.[6] It is important to realise that this comparison does not yet determine with which arguments a dispute can be won; it only tells us something about the relation between two individual arguments (and their subarguments).

To capture the different 'force' of the two ways of attack, we define 'defeat' in terms of two other notions, rebutting and undercutting[7] an argument. Their difference depends on whether the attacking argument contradicts a conclusion or an assumption of the attacked argument. Only in the first case will the priorities be used. This is based on our view that in legal reasoning the priorities are only used to resolve head–to–head conflicts between rules, not to resolve head–to–body conflicts. The definition of defeat will state how to use the notions of rebutting and undercutting an argument in combination.

Let us first concentrate on head–to–head conflicts and see how the priorities can be used to compare the conflicting arguments. In its simplest form, with a head–to–head conflict between two defeasible rules, we will compare the arguments with respect to the priority relation between these two rules (for a defence of this approach over comparing the arguments with respect to their 'weakest links', the reader is referred to (Prakken, 1991; Prakken, 1993)). However, if the conflict involves strict rules, then each argument may contain more than one rule of which the priority is relevant. To see this, suppose that $A = [r_1: \Rightarrow Rab, r_2: \Rightarrow Rbc]$, $B = [r_3: \Rightarrow Rca]$, and that S contains transitivity and asymmetry axioms for R. Then A can with these axioms be extended to an argument for $\neg Rca$, which means that A attacks B and vice versa. Now if we examine A, it seems natural to regard the priority of both r_1 and r_2 as relevant to deciding the conflict. So we want to compare r_3 with the set of rules $\{r_1, r_2\}$. In general, both arguments involved in a conflict can have more than one rule directly relevant to the conflict, and therefore in general we have to compare two *sets* of defeasible rules.

So what are the rules that are relevant for the comparison? In the simple case, when the conflict is between two defeasible rules with complementary consequents, they are exactly those defeasible rules. In the more complex case, when the conflict is determined by strict rules, they are, for each argument, all defeasible rules that are used to satisfy the antecedents of the strict rules with which the argument is (hypothetically) extended to establish a conflict. In the special case when the conflict is entirely caused by strict rules, the sets of relevant defeasible rules will be empty. Formally:

DEFINITION 4.14. For any argument A and set S of strict rules such that $A+S$ is an argument with conclusion L we define the set $R_L(A+S)$ of the defeasible rules relevant to L in the argument $A+S$ as follows. $R_L(A+S) =$

1. $\{r_d\}$ iff A includes a defeasible rule r_d with consequent L;
2. $R_{L_1}(A+S) \cup \ldots \cup R_{L_n}(A+S)$ iff A is defeasible and S includes a strict rule r_s: $L_1 \wedge \ldots \wedge L_n \rightarrow L$.

Observe that since by definition no argument contains two rules with the same consequent, the set $R_L(A+S)$ is always unique.

To illustrate this definition, in the examples 4.6 and 4.10 the rules to be compared are r_1 and r_2. Consider next $A = [r_1\colon \Rightarrow Rab,\ r_2\colon \Rightarrow Rbc,\ s_1\colon Rab \wedge Rbc \rightarrow Rac]$. Then $R_{Rac}(A) = \{r_1, r_2\}$. However, if we make s_1 a defeasible rule, then $R_{Rac}(A) = \{s_1\}$.

Next we define how sets of rules are ordered on the basis of an ordering on their members.

DEFINITION 4.15. For any two sets R and R' of defeasible rules, $R < R'$ iff for some $r \in R$ and all $r' \in R'$ it holds that $r < r'$.

The intuitive idea behind this definition is that if $R < R'$, R can be made better by replacing some rule in R with any rule in R', while the reverse is impossible.

Now we have enough tools to give the definition of rebutting and undercutting arguments. In fact, it will be almost the same as the one of attack; the only difference is that in the first case of attack, when arguments have complementary conclusions, we will use the priorities to assess them. Since we will not use the priorities in the second case, this means that an attack on an assumption always succeeds.

DEFINITION 4.16. Let A_1 and A_2 be two arguments. Then

- A_1 *rebuts* A_2 iff condition (1) of Definition 4.8 holds, provided that $R_L(A_1 + S_1) \not< R_{\overline{L}}(A_2 + S_2)$.

- A_1 *undercuts* A_2 iff condition (2) of Definition 4.8 holds.

To inspect some properties of the definition, note first that from the definition of attack these definitions inherit that also rebutting and undercutting an argument can be 'direct', or 'indirect'. Let us now investigate some other properties. Observe first that if A rebuts or undercuts B, then A attacks B. Obviously, it does not hold that if A undercuts B, then B undercuts A. However, neither does it hold that if A undercuts B, then B does not undercut A. A counterexample is $A = [r_1\colon \sim b \Rightarrow a], B = [r_2\colon \sim a \Rightarrow b]$. It is also not the case that if A rebuts B, then B rebuts A. Just assume we have $A = [r_1\colon \Rightarrow a]$, $B = [r_2\colon \Rightarrow \neg a]$, and $r_2 < r_1$. However, rebutting involving a $<$ relation between conflicting rules is not always one–directional. To see this, assume in Example 4.10 that $r_1 < r_4$ and $r_3 < r_2$; then $[r_1, r_2]$ and $[s_1, r_3, r_4]$ rebut each other.

Finally we can give our definition of defeat. It states how in evaluating arguments the notions of undercutting and rebutting attack are combined. As a border case it regards any incoherent argument as defeated by the empty argument. Apart from this, the definition is based on two ideas, of which the first is implicit, and inherited from the definitions of attack, rebutting and undercutting: defeat of an argument can be direct, or indirect, by defeating one of its strict subarguments. The other idea is that our reading of $\sim A$ not only makes attacks on assumptions always succeed, but also makes an attack on an assumption stronger than an attack on a conclusion: if one argument undercuts the other, and the other does not undercut but only rebuts the first, the first defeats the second but the second does not defeat the first. Below we will explain the reason for our choice with an example.

DEFINITION 4.17. Let A_1 and A_2 be two arguments. Then A_1 *defeats* A_2 iff A_1 is empty and A_2 is incoherent, or else if

- A_1 undercuts A_2; or

- A_1 rebuts A_2 and A_2 does not undercut A_1.

We say that A_1 *strictly defeats* A_2 iff A_1 defeats A_2 and A_2 does not defeat A_1.

To check some boundary cases, from Definition 4.15 we have that for any nonempty set of rules R, $R < \emptyset$; this makes strict arguments strictly defeat conflicting defeasible arguments. Moreover, we have that $\emptyset \not< \emptyset$, so that no strict argument strictly defeats any conflicting strict argument (recall that strict rules contain no weak literals).

The following example illustrates that undercutting an argument is stronger than rebutting it.

EXAMPLE 4.18. Consider

$$r_1: \quad \sim \neg\, OJ \text{ is innocent} \Rightarrow OJ \text{ is innocent}$$
$$r_2: \quad \Rightarrow \neg\, OJ \text{ is innocent}$$

And assume that $< \, = \emptyset$. Then, although $[r_1]$ rebuts $[r_2]$, $[r_1]$ does not defeat $[r_2]$, since $[r_2]$ undercuts $[r_1]$. So $[r_2]$ strictly defeats $[r_1]$.

With this example we can explain why we regard undercutting attack as stronger than rebutting attack. This is since the only way to accept both rules is to believe that OJ is not innocent: in that case the condition of r_1 is not satisfied. By contrast, if it is believed that OJ is innocent, then r_2 has to be rejected, in the sense that its antecedent is believed but its consequent is not. In fact, we here employ the logical counterpart of the legal principle that the law should be interpreted as coherently as possible.

If we examine the nature of our notion of defeat, we see that it is a weak one, since for rebutting an argument no $>$ relation is needed between the sets of relevant rules, but only a $\not<$ relation. So, rather than having to be 'really better' than the argument that is to be defeated, a defeating argument only has to be not inferior. The reason we have this weak notion of defeat is that we want that our notion of 'justified arguments', to be defined next, really regards only those arguments as justified that, given the premises, are beyond any reasonable doubt or challenge. And reasonable doubt can be cast on an argument just by providing a counterargument that is not inferior to it.

For the same reason we also need the notion of strict defeat, which, being asymmetric, captures the idea of 'really being better than'. In the following section this notion will be used to ensure that a counterargument fails to cast doubt if it is inferior to at least one counterargument that is itself protected from defeat.

5. The status of arguments

5.1. The general idea

Since defeating arguments can in turn be defeated by other arguments, comparing pairs of arguments is not sufficient; what is also needed is a definition that determines the status of arguments on the basis of all ways in which they interact. In particular, the definition should allow for reinstatement of defeated arguments, if their defeater is itself (strictly) defeated by another argument. Also, the definition should respect the 'weakest link' principle that an argument cannot be justified unless all of its subarguments are justified.

Defining this notion, then, is the central part of our system. This part takes as input the set of all possible arguments and their mutual relations of defeat, and produces as output a division of arguments into three classes: arguments with which a dispute can be 'won', respectively, 'lost' and arguments which leave the dispute undecided. As remarked above, the winning arguments should be only those arguments that, given the premises, are beyond reasonable doubt: the only way to cast doubt on these arguments is by providing new premises, giving rise to new, defeating counterarguments. Accordingly, we want our set of justified arguments to be unique, and to be conflict–free.

Inspired by earlier work of (Vreeswijk, 1993b; Dung, 1994; Brewka, 1994b; Loui & Norman, 1995) we develop this part of our framework as a proof theory in dialectical style, as a dialogue game. A proof that an argument is justified will take the form of a dialogue tree, where each branch of the tree is a dialogue, and the root of the tree is that argument. The idea is that every move in a dialogue consists of an argument based on the input theory, where each stated argument attacks the last move of the opponent in a way that meets the player's burden of proof. That a move consists of a complete argument, means that the search for an individual argument is conducted in a 'monological' fashion, determined by the nature of the underlying logic; only the process of considering counterarguments is modelled dialectically. The required force of a move depends on who states it. Since the proponent wants a conclusion to be justified, a proponent's argument has to be strictly defeating, while since the opponent only wants to prevent the conclusion from being justified, an opponent's move may be just defeating.

Before we illustrate and define these ideas, a remark is in order on the origin of the input theory. Logically, the dialogue game that we will define assumes a given 'pool' of rules from which the arguments are to be constructed. At first sight, this would seem to run counter to the way actual legal disputes proceed, where the parties can freely introduce new premises with every argument move. However, our assumption of a prior pool of premises is just a theoretical abstraction; nothing prevents applications to situations where the pool of premises consists of everything put forward by any of the parties in a dialogue.

Let us illustrate our ideas with a dialogue based on the following example.

EXAMPLE 5.1. Assume we have the following input theory.

r_1: $\sim\neg$ e is admissible evidence $\Rightarrow$ e proves guilt
r_2: $\Rightarrow$ f forged evidence e
r_3: x forged evidence $y \Rightarrow \neg$ y is admissible evidence
r_4: x is police officer $\wedge \sim x$ is dishonest $\Rightarrow \neg$ x forged evidence y
f_5: $\rightarrow$ f is police officer in LA
f_6: x is police officer in LA $\rightarrow$ x is police officer
r_7: x is police officer in LA $\Rightarrow$ x is dishonest
r_8: $\Rightarrow$ f has received a medal of honour
r_9: x is police officer $\wedge$ x has received a medal of honour $\Rightarrow$
$\neg$ f is dishonest

and assume that, for whatever reason, $r_2 < r_4$, $r_7 < r_9$ and that all other rules are incomparable. Let us denote the arguments stated by the proponent by P_i and those of the opponent by O_i. The proponent starts the dispute by asserting that $[r_1]$ is a justified argument for 'e proves guilt'.

P_1: [r_1: $\sim\neg$ e is admissible evidence $\Rightarrow$ e proves guilt]

Now the opponent has to defeat this argument. It can do so in only one way, by undercutting it with

O_1: [r_2: $\Rightarrow$ f forged evidence e,
r_3: f forged evidence $e \Rightarrow \neg$ e is admissible evidence]

The proponent now has to counterattack with an argument that strictly defeats O_1. It finds the following rebuttal, using the fact that $r_2 < r_4$.

P_2: [f_5: $\rightarrow$ f is police officer in LA,
f_6: f is police officer in LA $\rightarrow$ f is police officer,
r_4: f is police officer $\wedge \sim f$ is dishonest
$\Rightarrow \neg$ f forged evidence e]

Again the opponent has only one way to respond, with the undercutting attack:

O_2: [f_5: $\rightarrow f$ is police officer in LA,
r_7: f is police officer in LA $\Rightarrow f$ is dishonest]

This time the proponent responds with a strictly defeating rebuttal, using that $r_7 < r_9$.

P_3: [r_8: $\Rightarrow f$ has received a medal of honour,
f_5: $\rightarrow f$ is police officer in LA,
f_6: f is police officer in LA $\rightarrow f$ is police officer,
r_9: x is police officer $\wedge$ x has received a medal of honour $\Rightarrow \neg f$ is dishonest]

Now the opponent has run out of moves: no argument on the basis of our ordered theory defeats P's last argument. And since at no point could O create alternative branches, P can successfully defend its argument against every possible way of attack: r_1 is a provably justified argument.

In dialectical proof systems a 'loop checker' can be implemented in a very natural way, by stating that no two moves of the proponent in the same branch of the dialogue may have the same content.

EXAMPLE 5.2. Assume for illustration that our ordered theory Γ is the following familiar logic–programming example.

r_1: $\sim a \Rightarrow b$
r_2: $\sim b \Rightarrow a$

This is how the attempt to prove b fails.

P_1: [r_1: $\sim a \Rightarrow b$]
O_1: [r_2: $\sim b \Rightarrow a$]

Now P has run out of moves, since any defeating attack on O_1 would have the same content as P_1.

It is easy to see that a non–repetition rule will not harm P; if O had a move the first time P stated the argument, it will also have a move the second time, so no repetition by P can make P win a dialogue.

5.2. The dialogue game

We now formally define the dialogue game. Our aim is to make it agree with the fixpoint definition of (Prakken & Sartor, 1995; Prakken &

Sartor, 1996a), which is a variant of (Dung, 1993)'s grounded (skeptical) semantics of extended logic programming. More precisely, we want our dialogue game, when interpreted as a proof theory for this semantics, to be sound in the general case, and for justified arguments also complete for finitary ordered theories, i.e. for theories in which each argument has at most a finite number of attackers. For the proofs that this is the case we refer to (Prakken & Sartor, 1996b).

Justified arguments
We start with the notion of justified arguments.

DEFINITION 5.3. A *dialogue* based on an ordered theory Γ is a finite nonempty sequence of moves $move_i = (Player_i, Arg_i)$ $(i > 0)$, such that

1. $Arg_i \in Args_\Gamma$;
2. $Player_i = P$ iff i is odd; and $Player_i = O$ iff i is even;
3. If $Player_i = Player_j = P$ and $i \neq j$, then $Arg_i \neq Arg_j$;
4. If $Player_i = P$, then Arg_i is a minimal (w.r.t. set inclusion) argument strictly defeating Arg_{i-1} on the basis of Γ;
5. If $Player_i = O$, then Arg_i defeats Arg_{i-1} on the basis of Γ.

The first condition makes a dialogue relative to a given ordered theory, while the second condition says that the proponent begins and then the players take turns, and condition (3) prevents the proponent from repeating its attacks. The last two conditions form the heart of the definition: they state the burdens of proof for P and O. The minimality condition on P's moves makes it impossible to make arguments trivially different by combining them with some other, irrelevant argument.

In the remaining definitions we leave Γ implicit.

DEFINITION 5.4. A *dialogue tree* is a finite tree of moves such that

1. Each branch is a dialogue;
2. If $Player_i = P$ then the children of $move_i$ are all defeaters of Arg_i.

The second condition of this definition makes dialogue trees candidates for being proofs: it says that the tree should consider all possible ways in which O can attack an argument of P.

DEFINITION 5.5. A player *wins a dialogue* if the other player cannot move. And a player *wins a dialogue tree* iff it wins all branches of the tree.

The idea of this definition is that if P's last argument is undefeated, it reinstates all previous arguments of P that occur in the same branch of a tree, in particular the root of the tree.

DEFINITION 5.6. An argument A is *justified* iff there is a dialogue tree with A as its root, and won by the proponent.

Formal properties

In (Prakken & Sartor, 1996b) we have proven the desired results on soundness and completeness with respect to (Prakken & Sartor, 1996a). This means that every argument that is justified according to the dialogue game, is also justified according to the fixpoint semantics, while for finitary ordered theories every semantically justified argument is also dialectically justified. We have also shown that the set of justified arguments is unique and conflict–free, as desired. Finally, we have shown that our system respects the 'weakest link' principle that an argument cannot be justified if not all its subarguments are justified. Some systems incorporate this principle in the definition of a justified argument, e.g. (Vreeswijk, 1993a; Prakken, 1993; Sartor, 1993; Nute, 1994). In the present system this is not the case; instead it makes the principle follow from the other definitions, as also in e.g. (Pollock, 1992; Simari & Loui, 1992; Geffner & Pearl, 1992).

Overruled and Defensible Arguments

In (Prakken & Sartor, 1995; Prakken & Sartor, 1996a) the categories of overruled and defensible arguments are defined in a purely declarative way: an argument is *overruled* iff it is attacked by a justified argument, and it is *defensible* iff it is neither justified nor overruled. This definition is still possible in our present system, but then the question arises what are the corresponding dialogue rules. We will only discuss this for defensible arguments, since we do not expect that many parties in a dispute will want to defend their own argument as overruled.

So when is an argument dialectically shown to be defensible? The key idea is to reverse the burden of proof for O and P: the opponent now has to find an argument that *strictly* defeats the proponent's previous move, while the proponent only needs to find an argument that *defeats* the opponents last move. Moreover, the non–repetition condition now holds for the opponent instead of the proponent. For the rest the definitions remain unchanged: in particular, if the proponent can make the opponent run out of moves against every attacking strategy of the opponent, the argument is shown to be at least defensible.

Soundness of this definition with respect to (Prakken & Sartor, 1996a) is straightforward. Issues of completeness and the relation with other credulous semantics defined by (Dung, 1995), such as stable semantics, are left for future research. Here we confine ourselves to an example. Consider again the ordered theory of Example 5.2. Above we saw that P's attempt to prove P_1 to be justified failed. However, P can prove it to be defensible.

P_1: [r_1: $\sim a \Rightarrow b$]

And already O has run out of moves, since O_1, although defeating P_1, does not strictly defeat it, as is now required.

The Status of Conclusions
Since ultimately we are not interested in arguments but in the conclusions they support, we also have to define the status of strong literals. This can be done straightforwardly, by saying that L is a *justified conclusion* iff it is a conclusion of a justified argument; L is a *defensible conclusion* iff it is not justified and it is a conclusion of some defensible argument; and L is an *overruled conclusion* iff it is not justified or defensible, and a conclusion of an overruled argument.

5.3. Illustrations

EXAMPLE 5.7. The following example illustrates that our definitions respect the 'step–by–step' nature of argumentation: conflicts about conclusions or assumptions earlier in the chains are dealt with before 'later' conflicts.

r_1:	$\Rightarrow a$	r_2:	$a \Rightarrow b$	r_3:	$b \Rightarrow c$
r_4:	$\Rightarrow \neg a$	r_5:	$\neg a \Rightarrow d$	r_6:	$d \Rightarrow \neg c$

Assume that $< = \{r_4 < r_1,\ r_3 < r_6\}$. Then the arguments $[r_1 - r_3]$ and $[r_4 - r_6]$ defeat each other. However, the first argument can still be proven to be justified, because its subargument $[r_1]$ wins a conflict with a subargument of the second, viz. $[r_4]$. Here is the proof.

P_1:	[r_1:	$\Rightarrow a$,	r_2:	$a \Rightarrow b$,	r_3:	$b \Rightarrow c$]
O_1:	[r_4:	$\Rightarrow \neg a$,	r_5:	$\neg a \Rightarrow d$,	r_6:	$d \Rightarrow \neg c$]
P_2:	[r_1:	$\Rightarrow a$]				

O cannot counter with [r_4: $\Rightarrow \neg a$]: that argument does not defeat P_2, since $r_4 < r_1$.

EXAMPLE 5.8. Finally, we show that our system does not exhibit a form of 'extreme skepticism', to be found, for instance, in (Horty et al., 1990; Nute, 1994). Informally speaking, a system is extremely skeptical if arguments have to be 'cut off' not only when they are overruled, but also when they are merely defensible. In our system this is not the case, as is illustrated by the following example.

r_1: $\Rightarrow f$ forged evidence e
r_2: f forged evidence $e \Rightarrow \neg\ e$ is admissible evidence
r_3: $\Rightarrow \neg\ f$ forged evidence e
r_4: $\sim \neg\ e$ is admissible evidence $\Rightarrow e$ proves guilt

Assume that $< = \emptyset$. Here is how the proof for 'e proves guilt' fails.

P_1: [r_4: $\sim \neg\ e$ is admissible evidence $\Rightarrow e$ proves guilt]
O_1: [r_1: $\Rightarrow f$ forged evidence e,
r_2: f forged evidence $e \Rightarrow \neg\ e$ is admissible evidence]

Now P has run out of moves, since $[r_3]$ defeats O_1 only nonstrictly. By contrast, in extremely skeptical systems $[r_1, r_2]$ is, in our terms, not allowed to prevent $[r_4]$ from becoming justified, since it has a subargument that is only defensible. As a result, in such systems $[r_4]$ is justified, even though it is attacked by an argument that is not worse than any counterargument.

Although this difference reflects a 'clash of intuitions', we think that our intuitions are justified by our general approach. Recall that we regard an argument as justified only if, given the premises, no doubt at all can be cast on the argument. Now here doubt can be cast on the argument $[r_4]$, since it has a counterargument that is not weaker than any argument attacking it. We think that in a dispute a judge would feel compelled to determine whether $r_1 < r_3$ before deciding in favour of P.

5.4. The dialogue game: declarative or procedural?

In Section 2 we remarked that argumentation frameworks are not procedural but declarative, or relational theories, since their only job is to define a relation between premises and conclusions: given a body of input information they declare arguments to have a certain status. Now the dialogue game defined in the present section clearly has a procedural flavour; does this mean that we have shifted from a declarative to a procedural perspective?

We think that this is partly the case. Although the definitions of Section 4, together with the fixpoint characterisation of the justified arguments of (Prakken & Sartor, 1996a) clearly form an AF, the present dialogue game may be regarded as the logical core of a dialectical protocol, i.e. as that part of a protocol that applies the AF. Thus our dialogue game links the AF with a dialectical protocol. We can even state this more generally: our dialogue game links any theory that produces a binary relation of defeat among arguments with any dialectical protocol that incorporates our dialogue rules.

6. The formal system II: defeasible priorities

6.1. The new definitions

So far we have simply assumed that there is a fixed and undisputable ordering on the rules. However, as we already said in the introduction, this assumption is unrealistic: in legal reasoning, and also in several other domains of practical reasoning, the standards for conflict resolution are themselves subject to debate and disagreement. A full theory of defeasible argumentation in law should therefore also be able to formalise arguments about priorities.

Obviously, we cannot simply do this by only adding a priority predicate symbol to our object language; in addition the priorities derived at the object level somehow need to be lifted to the metatheory of the system, in particular to Definition 4.15. So we have to define a formal connection between object and metalevel.

We start by assuming that our language contains a distinguished twoplace predicate symbol $\prec$, with which information on the priorities can be expressed in the object language. This makes the third component of an ordered theory redundant, so an *ordered theory* is from now on just a pair (S, D). Next we have to make sure that the ordering thus derived is of the desired type, i.e. that it is a strict partial order. This can be done by adding the axioms of a partial order to the strict rules. A slight complication is that, since strict rules do not allow for contraposition, we also have to add the contrapositive rules for transitivity (for asymmetry the contrapositive is redundant). Thus, in the rest of this paper we assume that of every input theory the set S of strict rules contains all and only the following rules containing the $\prec$ predicate.

t_1: $x \prec y \wedge y \prec z \rightarrow x \prec z$
t_2: $x \prec y \wedge \neg\, x \prec z \rightarrow \neg\, y \prec z$
t_3: $y \prec z \wedge \neg\, x \prec z \rightarrow \neg\, x \prec y$
a: $x \prec y \rightarrow \neg\, y \prec x$

We now need to link the priority conclusions that can be drawn at the object level to the metalevel of our system: i.e. we want to make sure that $(r, r') \in <$ if and only if there is a justified argument for $r \prec r'$. We retain the idea of (Prakken & Sartor, 1995) and (Prakken & Sartor, 1996a) that the ordering component of an ordered theory Γ is now determined by the set of all priority arguments that are justified on the basis of Γ. Therefore, we have to define the following notation, capturing what a certain set of arguments says about the priorities.

DEFINITION 6.1. For any set *Args* of arguments

$$<_{Args} = \{r < r' \mid r \prec r' \text{ is a conclusion of some } A \in Args\}$$

And we say for any set of arguments *Args* that A (strictly) *Args*–defeats B on the basis of Γ iff according to Definition 4.17 A (strictly) defeats B on the basis of $(\Gamma, <_{Args})$. Occasionally, we will also use the analogous notion *Args*–rebuts. And for any argument A we write $\{A\}$–defeats as A–defeats.

Now we have to change our rules of the dialogue game. The main problem here is on the basis of which priorities the defeating force of the moves should be determined. What we want to avoid is that we have to generate all priority arguments before we can determine the defeating force of a move. The pleasant surprise is that, to achieve this, a few very simple conditions suffice. For O it is sufficient that its move $\emptyset$–defeats P's previous move. This is so since, as shown in (Prakken & Sartor, 1996b), Definition 4.17 implies that if A is for some S an S–defeater of P's previous move, it is also an $\emptyset$–defeater of that move. So O does not have to take priorities into account, as is illustrated by

P_1: $[r_1: \Rightarrow a]$

Now a possible move of O is

O_1: $[r_2: \Rightarrow b,\ \ r_3: b \Rightarrow \neg a]$

P, on the other hand, should take some priorities into account. However, it suffices to apply only those priorities that are stated by P's move;

more priorities are not needed, since Definition 4.17 also implies that if P's argument Arg_i strictly $Args_i$–defeats O's previous move, it will also do so whatever more priorities will be derived. So P can reply to O_1 with

P_2: $[r_4\colon \Rightarrow \neg b,\ \ r_5\colon \Rightarrow r_2 \prec r_4]$

However, this is not the only type move that the proponent can make. If O responds with

O_2: $[r_2\colon \Rightarrow b]$

which $\emptyset$–defeats P_2, then P must have the possibility to 'undo' the defeating force of O_2 with the priority argument

P_3: $[r_5\colon \Rightarrow r_2 \prec r_4]$

We now incorporate these ideas in the definition of a dialogue. All conditions are the same as above, except for the references to 'defeat'.

DEFINITION 6.2. A *priority dialogue* based on Γ is a finite sequence of moves $move_i = (Player_i, Arg_i)$ $(i > 0)$, where

1. $Arg_i \in Args_\Gamma$;
2. $Player_i = P$ iff i is odd, and $Player_i = O$ iff i is even;
3. If $Player_i = Player_j = P$ and $i \neq j$, then $Arg_i \neq Arg_j$;
4. If $Player_i = P$ then Arg_i is a minimal (w.r.t. set inclusion) argument such that
   - Arg_i strictly Arg_i–defeats Arg_{i-1} on the basis of Γ; or
   - Arg_{i-1} does not Arg_i–defeat A_{i-2} on the basis of Γ;
5. If $Player_i = O$ then Arg_i $\emptyset$–defeats Arg_{i-1} on the basis of Γ.

The only change in the definition of a dialogue tree is that the defeat condition on O's moves is made relative to the empty set.

DEFINITION 6.3. A *priority dialogue tree* is a finite tree of moves such that

1. Each branch is a dialogue;
2. If $Player_i = P$ then the children of $move_i$ are all $\emptyset$–defeaters of Arg_i.

The other definitions stay the same.

In (Prakken & Sartor, 1996b) we prove that the abovementioned formal properties of the system with strict priorities also hold for the defeasible case

6.2. Applications

In the remaining examples we will use the following method for naming rules. Every rule with terms $t_1, \ldots, t_n$ is named with a function expression $r(t_1, \ldots, t_n)$, where r is the informal name of the rule.

First we propose a systematic way of formalising reasoning about multiple legal orderings. The idea is to just write 'if' definitions, so that the ordering is possibly partially unspecified. The three general principles Lex Superior, Lex Specialis and Lex Posterior thus become:

$$\begin{array}{ll} H(x,y): & x \text{ is inferior to } y \Rightarrow x \prec y \\ T(x,y): & x \text{ is earlier than } y \Rightarrow x \prec y \\ S(x,y): & y \text{ is more specific than } x \Rightarrow x \prec y \end{array}$$

Other rules can then specify when the antecedents of these rules hold. For example,

$$\begin{array}{ll} r_1(r_3(x), r_4(x)): & \Rightarrow r_4(x) \text{ is more specific than } r_3(x) \\ r_2(x,y): & x \text{ is in a statute} \wedge y \text{ is in the constitution} \\ & \Rightarrow x \text{ is inferior to } y \end{array}$$

$r_1(r_3(x), r_4(x))$ says of two particular rules $r_3(x)$ and $r_4(x)$ that one is more specific than the other. This premise could be specified 'by hand' by the user but it could also be the outcome of a logical specificity checker, as e.g. used by (Simari & Loui, 1992).

Finally, the relation between the three principles can be specified by

$$\begin{array}{ll} O_1(T(x,y), H(x,y)): & \Rightarrow T(x,y) \prec H(x,y) \\ O_2(S(x,y), T(x,y)): & \Rightarrow S(x,y) \prec T(x,y) \end{array}$$

EXAMPLE 6.4. We now formalise an example in which the Italian priority rule on building regulations, mentioned in section 3, conflicts with the temporality principle that the later rule has priority over the earlier one. They state contradicting priorities between a town planning rule saying that if a building needs restructuring, its exterior may be modified, and an earlier, and conflicting, artistic-buildings rule saying that if a building is on the list of protected buildings, its exterior may not be modified.

Note that rule r_9 states that rule r_3 is later than the Lex Posterior principle T, which implies that r_3 prevails over T, according to T itself. The application of a priority rule to itself (or better, to one of its instances), is an interesting peculiarity of this example.

$r_1(x)$:	x is a protected building $\Rightarrow \neg$ x's exterior may be modified
$r_2(x)$:	x needs restructuring $\Rightarrow$ x's exterior may be modified
$r_3(x, y)$:	x is about the protection of artistic buildings $\wedge$ y is a town planning rule $\Rightarrow y \prec x$
$T(x, y)$:	x is earlier than $y \Rightarrow x \prec y$
$r_4(r_1(x))$:	$\Rightarrow r_1(x)$ is about the protection of artistic buildings
$r_5(r_2(x))$:	$\Rightarrow r_2(x)$ is a town planning rule
$r_6(r_1(x), r_2(y))$:	$\Rightarrow r_1(x)$ is earlier than $r_2(y)$
$r_7(Villa_0)$:	$\Rightarrow Villa_0$ is a protected building
$r_8(Villa_0)$:	$\Rightarrow Villa_0$ needs restructuring
$r_9(T(x, y), r_3(x, y))$:	$\Rightarrow T(x, y)$ is earlier than $r_3(x, y)$

To maintain readability, we will below only give the function–symbol part of the rule names. Here is a proof that the exterior of $Villa_0$ may not be modified.

P_1: [r_7: $\Rightarrow Villa_0$ is a protected building,
r_1: $Villa_0$ is a protected building $\Rightarrow \neg$ $Villa_0$'s exterior may be modified]

O can respond in only one way.

O_1: [r_8: $\Rightarrow Villa_0$ needs restructuring,
r_2: $Villa_0$ needs restructuring $\Rightarrow Villa_0$'s exterior may be modified]

P can now neutralize O's defeater with the following priority argument, saying that the protection rule prevails over the town planning rule.

$$
\begin{array}{lll}
P_2\!: & [r_4\!: & \Rightarrow r_1 \text{ is about the protection of artistic buildings,}\\
 & r_5\!: & \Rightarrow r_2 \text{ is a town planning rule,}\\
 & r_3\!: & r_1 \text{ is about the protection of artistic buildings} \wedge\\
 & & \;r_2 \text{ is a town planning rule} \Rightarrow r_2 \prec r_1]
\end{array}
$$

But O can $\emptyset$–defeat this priority argument with a conflicting priority argument based on the Lex Posterior principle.

$$
\begin{array}{lll}
O_2\!: & [r_6\!: & \Rightarrow r_1 \text{ is earlier than } r_2,\\
 & T\!: & r_1 \text{ is earlier than } r_2 \Rightarrow r_1 \prec r_2]
\end{array}
$$

Now P takes the debate to the meta–meta–level, by asserting a priority argument that neutralizes O's defeater at the meta–level. P's argument says that, since the Lex Posterior principle is earlier than the building regulations principle, the former is inferior to the latter on the basis of the former. Although this seems to be self–referential, formally it is not, since it is one instance of Lex Posterior that speaks about another instance of itself.

$$
\begin{array}{lll}
P_3\!: & [r_9\!: & \Rightarrow T \text{ is earlier than } r_3,\\
 & T'\!: & T \text{ is earlier than } r_3 \Rightarrow T \prec r_3]
\end{array}
$$

Now O has run out of moves, and we know that the villa's exterior may not be modified. Note that in this dialogue the function arguments of T and r_3 are instances of r_1 and r_2, while of T' they are the full versions of T and r_3. We leave it to the computer to write the full name of T'.

EXAMPLE 6.5. Consider next the pasta example from the introduction, which combines priorities and assumptions.

$$
\begin{array}{ll}
r_1(x,y,z)\!: & x \text{ can be sold in } y \wedge\\
 & y \text{ is an E.C. country} \wedge\\
 & z \text{ is an E.C. country} \wedge\\
 & \sim \text{the selling of } x \text{ in } z \text{ endangers public health} \wedge\\
 & \sim \text{the selling of } x \text{ in } z \text{ prejudices the consumer} \Rightarrow\\
 & x \text{ can be sold in } z\\
r_2(x, Italy)\!: & x \text{ is called ``pasta''} \wedge\\
 & \neg x \text{ is made of hard corn} \Rightarrow\\
 & \neg x \text{ can be sold in } Italy
\end{array}
$$

Let us assume the following facts

$f_3(BRD)$:	$\rightarrow BRD$ is an EC–country
$f_4(Italy)$:	$\rightarrow Italy$ is an EC–country
$f_5(P_0, BRD)$:	$\rightarrow P_0$ can be sold in BRD
$f_6(P_0)$:	$\rightarrow P_0$ is called "pasta"
$f_7(P_0)$:	$\rightarrow \neg P_0$ is made of hard corn

We can then derive, as the European Court of Justice did, that P_0 can be sold in $Italy$, by means of the justified argument $A_1 = [f_3, f_4, f_5, r_1]$, which strictly defeats the rebutting argument $A_2 = [f_6, f_7, r_2]$. However, if we also ascertain that:

$r_8(P_0, Italy)$: $\Rightarrow$ the selling of P_0 in $Italy$ prejudices the consumer

then argument A_1 is undercut by argument $A_3 = [r_8]$, so that the before overruled argument B_2 for "$\neg P_0$ can be sold in $Italy$" becomes justified.

7. Applicability

Finally, we discuss the formalisation of statements about applicability. In legal language weak exceptions to rules are often made with a phrase like 'in these circumstances rule r is inapplicable'. Instead of attacking a particular assumption of a rule, such a phrase says that in these circumstances the whole rule may not be used for constructing arguments, in other words, that the rule is not a premise.

Basically, there are two methods for formalising applicability arguments. The first does not require a change of the definitions but complicates the representations of defeasible rules, while the second involves a change of the notions of attack and defeat, but leads to simpler representations.

The first method essentially boils down to the use of standard AI techniques for reasoning with 'abnormality' or 'applicability' predicates. For instance, if one has the view that any rule can make any other rule applicable, regardless of their preference relation, then it suffices to give each rule r an extra condition $\sim \neg appl(r)$. Then any argument with a conclusion $\neg appl(r)$ undercuts an argument using r.

Alternatively, if one thinks that a rule can only make another rule inapplicable if the first is not of lower priority, then to every defeasible rule r a condition $appl(r)$ can be added, and to D a new rule r_0 should be added of the form

$r_0(r)$: $\Rightarrow appl(r)$

Note that r here is a variable. Then arguments with defeasible rules can only be constructed by including instantiations of $r_0(r)$ for these rules. If now an argument has a conclusion $\neg appl(r_i)$, it does not attack an assumption but a conclusion of an argument using r_i and so needs priorities to succeed.

Here is how section 2 HPW, discussed in the introduction, can be formalised in this method. Recall that this section declares all other sections in the Dutch rent act inapplicable in case a lease is short–termed.

$$\begin{array}{ll} 2HPW(x,y): & appl(2HPW(x,y)) \wedge \\ & x \text{ is a short–termed lease } \wedge \\ & y \text{ is a section of the rent act } \wedge \\ & y \neq 2HPW(x,y) \Rightarrow \\ & \neg appl(y) \end{array}$$

Moreover, $2HPW$ should be given priority over the instances of r_0 for every other rule in the HPW. For more illustrations of this method, see (Prakken, 1993).

The second method, which was initially developed by (Sartor, 1994), involves a change of our formal system. The idea is to define inapplicability arguments as an additional way of attacking an argument: A attacks B also if B contains a defeasible rule r such that A has a conclusion $\neg appl(r)$. Note that thus rules are assumed to be applicable. Also (Hage, 1995) makes applicability a condition of the metatheory (of a system which is not argument– but extension–based), but he does not assume rules to be applicable; instead in his system a rule can only be applied if a conclusion that it is applicable is derivable.

Here is how the definitions of attack and defeat should be changed. First we add to Definition 4.8 a third condition to the disjunction:

(3) $A_2 + S_2$ is an argument with for some $r \in A_1$ a conclusion $\neg appl(r)$.

Then we add to Definition 4.16 a third possibility, *excluding* an argument by attacking the applicability of a rule in that argument. Again there is a choice whether the priorities should be relevant. If not, the following condition suffices.

(3) A_1 *excludes* A_2 iff condition (3) of Definition 4.8 holds.

Otherwise, we should add that the relevant rules for $\neg appl(r_1)$ are not all inferior to r_1.

(3) A_1 *excludes* A_2 iff condition (3) of Definition 4.8 holds, provided that $R_{\neg appl(r_1)}(A_1) \not< \{r_1\}$.

Finally, we redefine defeat as

DEFINITION 7.1. Let A_1 and A_2 be two arguments. Then A_1 *defeats* A_2 iff A_1 is empty and A_2 is incoherent, or else if

- A_1 undercuts A_2; or
- A_1 rebuts or excludes A_2 and A_2 does not undercut A_1.

With this definition it is not necessary to include applicability conditions in defeasible rules.

8. Some remarks on implementation

Our dialogue game already has the form of the top level of an implementation. Moreover, we have stated some results on completeness: if the input theory is finitary, i.e. it is such that each argument has at most a finite number of attackers, each argument that is semantically justified according to (Prakken & Sartor, 1996a) has a dialectical proof that it is justified. However, it should be noted that this is a kind of completeness that does not imply semi–decidability: if the language contains function symbols, the logic for the construction of arguments will not be decidable, and then the search for counterarguments is not even semi–decidable. So in the finitary case there is no algorithm that is guaranteed to find all proofs that exist.

In the case that an ordered theory has only a finite number of rules, decidability follows from the fact that, since arguments may not contain more than one rule with the same conclusion, a finite ordered theory has only a finite number of arguments. Note that since rules are schemes for all its instances, this condition implies that the language has no function symbols. At first sight, the exclusion of function symbols would seem problematic, since our just–explained naming convention makes heavily use of them. Yet we expect that in many practical applications also a simpler naming convention, with terms instead of function symbols, will do.

9. Comparison with related work

9.1. Logic and general AI

With respect to related research, we first indicate what we add to the earlier developments in defeasible argumentation, mentioned in section 2, in particular to (Loui, 1987) and (Simari & Loui, 1992). What we retain is the use of one–direction rules for representing conditionals, but we add assumptions, we replace specificity with any possible source of priorities and, finally, we make the priorities defeasible.

Next we focus on some logic–programming–like systems for defeasible reasoning. (Dung, 1993) has developed various argumentation based semantics of extended logic programming, based on more abstract work of (Dung, 1995) and (Bondarenko et al., 1993). As already indicated above, in (Prakken & Sartor, 1995) and (Prakken & Sartor, 1996a) we build on one of (Dung, 1993)'s definitions, in particular on his formulation of well–founded semantics of extended logic programs. However, there are some differences. Firstly, Dung does not consider reasoning with or about priorities. Moreover, since Dung only reformulates the various semantics of logic programs, in the case of well–founded semantics he inherits some of its features that we find less attractive. In particular, Dung defines, like (Bondarenko et al., 1993), arguments as being the set of assumptions of what we define as an argument. Thus the program r_1: $\Rightarrow a$, r_2: $\Rightarrow \neg a$, r_3: $\Rightarrow b$ has only one argument, viz. $\emptyset$, which defeats itself, so that the set of justified arguments is empty. By contrast, in our system we have a justified argument $[r_3]$ and two defensible arguments $[r_1]$ and $[r_2]$. This problem occurs in any program with head–to–head conflicts between rules.

Dung's assumption–based approach can be defended by saying that it determines sets of assumptions (i.e. weak literals) with which a program or theory can be given a noncontradictory interpretation. If no such set exists, the program is in some sense incoherent and the system draws no justified conclusions. Our system, however, is also meant to be used in adversarial contexts, where coherency is rare.

An early application of extended logic programming in which head–to–head conflicts between rules do not trivialise a program is (Kowalski & Sadri, 1990). In their method, rules with negative consequents are interpreted as exceptions to rules with the corresponding positive consequent. This reading is preserved by a translation of the extended program into a general logic program, i.e. a program with only weak negation. This method still has some limitations, such as the inability to express exceptions to exceptions.

In the case of fixed priorities our system is a variant of (Dung, 1994)'s dialogue game formulation of the well–founded semantics of (Dung, 1993). Our main contribution to Dung's version is that we extend it to the case of defeasible priorities. Already (Rescher, 1977) suggested the dialectical form to analyze defeasible reasoning. (Brewka, 1994b) gives a logical reconstruction of Rescher's suggestion with the help of default logic. (Vreeswijk, 1993b) has developed a dialectical proof theory for his own system for defeasible argumentation developed in (Vreeswijk, 1993a). Finally, Loui has in various publications defended a dialectical approach to the formalisation of defeasible reasoning, e.g. in (Loui, 1993).

(Brewka, 1996) has developed a version of well–founded semantics with defeasible priorities which is not argument–based. His version partly avoids the just–discussed problems, viz. in cases where the rules that are in a head–to–head conflict assume their head in their body. An important difference with our system is that Brewka reduces rebutting to undercutting defeat, by assuming of every rule that is intended to be defeasible its consequent in its antecedent. Although this makes the definitions simpler, it also gives up some expressive power. In particular, the difference between defeasible rules like 'If A then B' and 'If A then B, unless the contrary can be shown' cannot be expressed. On the other hand, it makes the priorities also applicable to mutually undercutting arguments, which Brewka finds desirable.

Also two other logic–programming–like systems are relevant, 'ordered logic' (Laenens & Vermeir, 1990)) and (Nute, 1994)'s 'defeasible logic'. Both systems have prioritized rules but not assumptions, while the priorities are fixed. They also both reflect the kind of skepticism discussed above with Example 5.8. Nute's language also has a 'might conditional', which roughly is the conditional denial of a rule. To obtain decidability, Nute assumes the input theory to be finite, and the language to have no function symbols.

Also, Nute's system has, like ours, the category of strict rules. However, Nute's treatment of such rules is different, since he regards only rules with complementary consequents as conflicting. Thus in the examples 4.5 and 4.6 he regards the arguments $[s_1, r_1]$ and $[s_2, r_2]$ as not conflicting, for which reason he does not take the priority relation between r_1 and r_2 into account when deciding the conflicts between the strict and defeasible arguments.

Finally, an alternative approach to reasoning about priorities, especially suitable for extension– or model–based nonmonotonic logics, was developed by (Brewka, 1994a) for prioritised default logic, and generalised and slightly revised by (Prakken, 1995a) for any extension– or model–based system. Roughly, the method checks whether what an extension says about the priorities corresponds to the ordering on which the extension is based.

9.2. AI AND LAW

We now discuss some projects in AI and Law. Here we only focus on relational, or declarative approaches. Most systems below are not formal systems but implemented architectures. As far as they share features with our framework, our work can be regarded as formulating standards for their behaviour.

First we discuss some systems using techniques of logic metaprogramming. (Yoshino & Kithahara, 1988) propose a rule–based system in which priorities between rules are defined on the basis of the principles Lex Superior, Lex Posterior and Lex Specialis, in that order of precedence. These rules are not represented in the knowledge representation language but encoded in the inference engine. Thus the relation between these three priority rules is fixed, other sources of priorities cannot be accounted for, and reasoning about priorities cannot be modelled. In (Yoshino, 1995) a more flexible rule–based inference engine is proposed, which accepts priorities as part of the input information. More exactly, each single priority criterion (such as Lex Posterior, Lex Superior, or Lex Specialis) contributes to the definition of the derogation relation. For example, later rules derogate older ones, and higher rules derogate lower ones. The inference engine checks the validity of every rule it uses, by checking that the rule is not derogated. The output of the system is a single category of conclusions, viz. those that can be drawn from valid rules. The system also checks the validity (i.e., non–derogation) of derogation rules, and this implies, in a way, reasoning about priorities. However, Yoshino does not address the problem of circular derogation statements, and he also leaves it unclear whether derogation statements only apply to rules that are conflicting or to all rules in the knowledge base.

(Hamfelt & Barklund, 1989) address, among other things, reasoning with prioritized legal rules. For this purpose they use the DEMO predicate of logic metaprogramming instead of explicit priorities, which makes their style of representation very different from ours: each preference rule is represented as a metaprogram, which outputs the conclusion of the weaker rules only if the conclusion of the conflicting stronger rules is not derivable. Moreover, in this approach, each set of rules having the same priority, under a determinate ordering criterion, is a separate program, which implies that a hierarchy of separated levels of legal knowledge is assumed. We have chosen not to make this assumption, because, as Example 6.4 shows, a conflict resolution rule may apply not only to object level rules, but also to conflict resolution rules, including itself.

Also (Routen, 1989) has suggested the use of logic metaprogramming, in particular, for dealing with rules and exceptions. However, Routen does not directly address the problem of reasoning with or about priorities.

The Helic system of (Nitta et al., 1995) is a software tool for legal reasoning, including an 'argumentation function', which builds arguments, and a 'debating function', which compares and evaluates them, implementing the model proposed in (Prakken, 1991; Prakken, 1993) and (Sartor, 1993; Sartor, 1994). The input of the system includes legal rules and priorities, which are called viewpoints. An example of a viewpoint is that flexibility in law is more important than strict interpretation of penal rules. A restriction is that viewpoints can only be stated as unconditional facts, while in our system they can be conditional upon other facts, just as any other rule.

Finally, (St–Vincent et al., 1995) use logic metaprogramming to define a filtering mechanism for building arguments, which selects the applicable rules by taking into consideration the contexts associated to them. A context is a set of clauses, representing, for instance, a set of facts, a legal text (such as a statute) or a category of legal texts (such as statutes or municipal regulations). Also in this framework a formalisation of priority rules is presented but the priority conclusions derived from them are not used to solve conflicts and evaluate arguments: the exclusive object of this work is argument construction.

Let us now pass to some other relational models of legal argumentation. Some of them are purely formal, while others implement a formal or informal theory of adversarial argument.

(Farley & Freeman, 1995) describe a dialectical model of legal reasoning with a heuristic flavour. They maintain that different legal problem solving contexts require different levels of proof. For instance, for the question whether a case can be brought before court, only a 'scintilla of evidence' is required, while for a decision in a case 'dialectical validity' is needed. They correspondingly describe a dialogue game in which parties dialectically construct arguments in order to satisfy (or to make unsatisfied) the required level of proof. The basic idea of their model is the same as of our dialogue game: only those moves are allowed that meet a certain level of burden of proof. However, there are also differences.

Firstly, Farley & Freeman divide the rules not in two but three epistemic categories: 'sufficient', 'evidential' and 'default', in decreasing order of priority. All three types of rules are in our sense defeasible, since they can all make arguments subject to defeat.

Farley & Freeman extend our model in that they also allow arguments based on modus tollens, and even nondeductive arguments, viz. simple forms of abductive and a contrario reasoning, with corresponding ways to attack such arguments. A subtlety is that nondeductive arguments can be disallowed by defining a certain level of burden of proof. Because of the presence of nondeductive arguments, arguments are not only compared on the strength of their rules, but also on the intrinsic strength of their reasoning steps.

Some aspects of Farley & Freeman's approach can be directly captured in our framework: for instance, their 'scintilla of evidence' corresponds to a defensible argument, while a dialectically valid proof corresponds to a justified argument. In general, their various levels of burden of proof can be captured by varying the required force of arguments in our dialogue game. On some other aspects our model generalises Farley & Freeman's approach, for instance, by including strict rules and weak negation. Also, rather than having to classify defeasible rules according to their epistemic nature, we can base the rule priorities on any possible ground, and we model reasoning about these priorities. Again other features are left outside our present framework, such as non-deductive arguments; since we want a completely formalised model, we prefer to study such arguments better, before including them in our system. Finally, as far as the inference rule of modus tollens for defeasible rules is concerned, we prefer not to introduce it into our system, for reasons extensively discussed in (Prakken, 1991; Prakken, 1993).

In various publications Hage has stressed the need for combining reasoning with and about priorities and about applicability; in fact, he reduces priority statements to applicability statements (cf. Yoshino's reduction to validity statements). A difference on details is that Hage does not study reasoning with assumptions. Also his general approach is different: technically since, like default logic, it is an extension–based system instead of an argument–based system; and philosophically, since it is inspired by (Raz, 1975)'s theory of reasons in practical reasoning. The latest version is described in (Hage, 1995).

One important issue raised by Hage is that of weighing reasons. He objects to systems like ours, where the defeat relation is defined between individual arguments. Instead, Hage wants to compare *sets* of arguments, to model the fact that in practical reasoning often reasons that individually have not enough weight to support a certain conclusion, do have that weight in combination. For instance, it might be that the individual reasons that it is hot, and that it is raining, are insufficient for not going jogging, but that they are sufficient in combination.

We agree that weighing reasons is an important feature of practical reasoning, but we prefer not to model it as an aspect of the assessment of arguments. Instead, we propose to express the weight of reasons in terms of priority relations between rules. For instance, the individual rules 'If it is hot, I don't go jogging' and 'if it is raining, I don't go jogging' can be given lower priority than 'If I haven't been jogging for two days, I go jogging'. And if the combination of the first two reasons outweighs the latter reason, this can be expressed by adding a new rule 'If it is hot and raining, I don't go jogging' and giving this rule higher priority than the jogging rule (note that also in Hage's system the information that a set of reasons outweighs another set has to be explicitly added).

The main reason why we prefer this approach is that reasons for a certain proposition cannot always be combined. For instance, for a certain person it can be that the individual circumstances that it rains and that it is hot are reasons for not going jogging, but that the combination is so pleasant that it is instead a reason for going jogging. Therefore, we don't want to be forced to regard the combination of two reasons for a certain conclusion also a reason for that conclusion, as is the case in Hage's approach. And precisely because this freedom to combine or not exists, we think that a decision to combine them in a particular case has the character of stating a new premise.

Next we discuss Gordon's work in (Gordon, 1994; Gordon, 1995). Part of his dialectical protocol for civil pleading is what can be called a declarative argumentation framework, based on (Geffner & Pearl, 1992)'s logic for conditional entailment. Within full first–order logic, Geffner and Pearl define a preferred–model semantics for any ordered set of defeasible conditionals, and then they define a sound and complete proof theory for this semantics. An important requirement on the ordering on defaults is 'admissibility', which in essence is that more specific defaults should have priority over more general ones. Gordon adds to Geffner & Pearl's theory of admissible orderings a method for expressing any given ordering (an issue not discussed by Geffner and Pearl), and also describes a method for representing explicit ordering information. In fact, his system has all the elements of the present framework: assumptions, applicability, strict and defeasible rules, and reasoning with and about priorities.

Still, there are some differences. In our system, the derived priorities are directly 'transported' to the metatheory. Since the predicate symbol $\prec$ in our object language denotes the relation $<$ in our metatheory. In Gordon's method, instead, priority statements are linked to priorities of conditional entailment's metatheory in a less transparent way. This is so since in conditional entailment specificity is hardwired in the logic (via the requirement that orderings are admissible), so that more specific rules always prevail. Therefore Gordon needs to transform winning rules (according to any priority criterion) into more specific ones. Obviously, since such a transformation does not succeed when the losing rule is more specific than the winning one, specificity becomes the overriding kind of defeat. As stated above, and as Gordon acknowledges in (Gordon, 1995), this is in legal applications undesirable: in the law, any priority relation is subject to debate.

10. Conclusion

To summarise the main results of the present paper, inspired by the rich structure of legal language and legal argument we have provided a formal argument–based system for the combination of several features of legal language: reasoning with priorities, reasoning about priorities and reasoning with assumptions and applicability statements. We have stated the system in a dialectical form, which easily suggests possible implementations. We have also shown that the system can be used to formalise a wide range of realistic examples from legal reasoning.

Several issues for future research remain. Firstly, on some points alternative definitions may be possible. For instance, examples might be found which motivate a different relation between rebutting and undercutting arguments, or different dialogue moves. However, we think that our system is robust enough to allow for such adjustments without having to change its overall structure.

We also want to investigate the use of our framework, in particular its dialectical features, for representing precedents in HYPO–style case–based reasoners. Here we want to explore an idea of (Loui & Norman, 1995) to represent precedents as argument structures, i.e. as a set of possibly conflicting arguments, including arguments on the resolution of such conflicts. We think that this is a promising application of argumentation frameworks, which can shed further light on the relation between rule–based and case–based theories of legal argument.

Acknowledgements

The authors thank Mark Ryan for his comments on an earlier version of this paper, the SG1 Working Group of ModelAge, led by Pierre–Yves Schobbens, for useful discussions, and the referees for their interesting comments.

Notes

[1] With 'procedure' and 'procedural' we refer to procedural law and not to logical or computational procedures.

[2] Where there is no danger of confusion, we will leave 'based on $(S, D, <)$' implicit.

[3] Sometimes we will leave implicit that we refer to ground instances of rules.

[4] Since the order of rules does not matter except that the antecedents of a rule should come from a previous rule, we ignore the fact that $A + T$ is not uniquely defined.

[5] To prevent terminological confusion, it should be noted that (Dung, 1993) and (Bondarenko et al., 1993) use the term 'attack' in a different way, coming closer to our notion of 'defeat', to be defined below. They do not have an explicit notion of what we call 'attack'.

[6] Note that the argument relations and properties defined below are relative to an implicitly assumed ordered theory.

[7] As further explained in (Prakken & Sartor, 1996b), our notion of undercutting is a special case of (Pollock, 1992)'s 'undercutting defeaters'.

References

Ashley, K.D. 1990. *Modeling legal argument: reasoning with cases and hypotheticals.* Cambridge, MA: MIT Press, 1990.

Bondarenko, A., Toni, F. & Kowalski, R.A. 1993. An assumption–based framework for non–monotonic reasoning. *Proceedings of the Second International Workshop on Logic Programming and Nonmonotonic Reasoning*, 1993, 171–189. Lisbon: MIT Press.

Brewka, G. 1994a. Reasoning about priorities in default logic. *Proceedings AAAI–94*, 247–260. Seattle.

Brewka, G. 1994b. A logical reconstruction of Rescher's theory of formal disputation based on default logic. *Proceedings of the 11th European Conference on Artificial Intelligence*, 366–370. Amsterdam: Wiley.

Brewka, G. 1996. Well–founded semantics for extended logic programs with dynamic preferences. *Journal of Artificial Intelligence Research* 4, 19– 36.

Dung, P.M. 1993. An argumentation semantics for logic programming with explicit negation. *Proceedings of the Tenth Logic Programming Conference*, 1993, 616–630. MIT Press.

Dung, P.M. 1994. Logic programming as dialogue games. Unpublished paper.

Dung, P.M. 1995. On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming, and n–person games. *Artificial Intelligence* 77, 321–357.

Farley, A.M. and Freeman, K. 1995. Burden of Proof in Legal Argumentation. *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*, 156–164. College Park, MD: ACM Press.

Geffner, H. and Pearl, J. 1992. Conditional entailment: bridging two approaches to default reasoning. *Artificial Intelligence* 53, 209–244.

Gordon, T.F. 1994. The Pleadings Game: An Exercise in Computational Dialectics. *Artificial Intelligence and Law* 2: 239–292.

Gordon, T.F. 1995. *The Pleadings Game. An Artificial Intelligence Model of Procedural Justice.* Kluwer, 1995.

Hage, J.C. Teleological reasoning in reason–based logic. *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*, 11–20. College Park, MD: ACM Press.

Hage, J.C., Leenes, R. & Lodder, A.R. 1994. Hard cases: a procedural approach. *Artificial Intelligence and Law* 2: 113–167.

Hamfelt, A. & Barklund, J. 1989. Metalevels in legal knowledge and their runnable representation in logic. *Preproceedings of the III International Conference on "Logica, Informatica, Diritto"*, Vol. II, 557–576. Florence.

Horty, J.F., Thomasson, R.H. & Touretzky, D.S. 1990. A skeptical theory of inheritance in nonmonotonic semantic networks. *Artificial Intelligence* 42, 311–348.

Kowalski, R.A. & Sadri, F. 1990. Logic programs with exceptions. *Proceedings of the Seventh International Logic Programming Conference*, 598–613. MIT Press.

Laenens. E. & Vermeir, D. 1990. A fixed points semantics for ordered logic. *Journal of Logic and Computation* Vol. 1 No. 2, 159–185.

Lodder, A.R. & Herczog, A. 1995. DiaLaw: A dialogical framework for modelling legal reasoning. *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*, 146–155. College Park, MD: ACM Press.

Loui, R.P. 1987. Defeat among arguments: a system of defeasible inference. *Computational Intelligence* 2, 100–106.

Loui, R.P. 1993. Process and policy: resource–bounded non–demonstrative reasoning. Report WUCS–92–43, Washington–University–in–St–Louis, 1993. To appear in *Computational Intelligence*.

Loui, R.P. & Norman, J. 1995. Rationales and argument moves. *Artificial Intelligence and Law* 3: 159–189.

Loui, R.P., Norman, J., Olson, J. & Merrill, A. 1993. A design for reasoning with policies, precedent, and rationales. *Proceedings of Fourth International Conference on Artificial Intelligence and Law*, 202–211. Amsterdam: ACM Press.

McCarty, L.T. & Sridharan, N.N. 1982. A computational theory of legal argument. Technical Report LRP–TR–13, Laboratory for Computer Science Research, Rutgers University, 1982.

Nitta, K. et al. 1995. New HELIC–II: a software tool for legal reasoning. *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*, 287–296. College Park, MD: ACM Press.

Nute, D.N. 1994. A decidable quantified defeasible logic. In D. Prawitz, B. Skyrms and D. Westerståhl (eds.): *Logic, Methodology and Philosophy of Science IX.* Elsevier Science B.V., 1994, 263–284.

Pollock, J.L. 1992. How to reason defeasibly. *Artificial Intelligence* 57, 1–42.

Prakken, H. 1991. A tool in modelling disagreement in law: preferring the most specific argument. *Proceedings of the Third International Conference on Artificial Intelligence and Law*, 165–174. Oxford: ACM Press.

Prakken, H. 1993. *Logical tools for modelling legal argument.* Doctoral Dissertation Free University Amsterdam, 1993.

Prakken, H. 1995a. A semantic view on reasoning about priorities (extended abstract). *Proceedings of the Second Dutch/German Workshop on Nonmonotonic Reasoning*, 152–159. Utrecht.

Prakken, H. 1995b. From logic to dialectics in legal argument. *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*, 165–174. College Park, MD: ACM Press.

Prakken, H. & Sartor, G. 1995. On the relation between legal language and legal argument: assumptions, applicability and dynamic priorities. *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*, 1–9. College Park, MD: ACM Press.

Prakken & Sartor, G. 1996a. A system for defeasible argumentation, with defeasible priorities. *Proceedings of the International Conference on Formal Aspects of Practical Reasoning*, Springer Lecture Notes in AI 1085, 510–524. Bonn: Springer Verlag.

Prakken & Sartor, G. 1996b. Argument–based extended logic programming with defeasible priorities. To appear in *Journal of Applied Non–classical Logics.*

Raz, J. 1975. *Practical reason and norms.* Princeton University Press, 1975.

Rescher, N. 1977. *Dialectics: a controversy–oriented approach to the theory of knowledge.* State University of New York Press, Albany, 1977.

Rissland, E.L. & Skalak, D.B. 1991. CABARET: statutory interpretation in a hybrid architecture. *International Journal of Man–Machine Studies* 34, 839–887.

Routen, T. 1989. Hierarchically organised formalisations. *Proceedings of the Second International Conference on Artificial Intelligence and Law*, 242–250. Vancouver: ACM Press.

Sartor, G. 1993. A simple computational model for nonmonotonic and adversarial legal reasoning. *Proceedings of the Fourth International Conference on Artificial Intelligence and Law*, 192–201. Amsterdam: ACM Press.

Sartor, G. A formal model of legal argumentation. *Ratio Juris* 7, 212–226.

Simari, G.R. & R.P. Loui, R.P. 1992. A mathematical treatment of defeasible argumentation and its implementation. *Artificial Intelligence* 53, 125–157.

Skalak, D.B. & Rissland, E.L. 1992. Arguments and Cases. An Inevitable Intertwining. *Artificial Intelligence and Law*, 3–44.

St–Vincent, P. Poulin, D. & Bratley, P. 1995. A Computational Framework for Dialectical Reasoning. *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*, 137–145. College Park, MD: ACM Press.

Vreeswijk, G. 1993a. *Studies in defeasible argumentation.* Doctoral dissertation Free University Amsterdam, 1993.

Vreeswijk, G. Defeasible dialectics: a controversy–oriented approach towards defeasible argumentation. *Journal of Logic and Computation*, Vol. 3, No. 3, 317–334.

Yoshino, H. 1995. The systematization of legal meta–inference. *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*, 266–275. College Park, MD: ACM Press.

Yoshino, H. and M. Kithahara, M. 1988. Les–Project. In H. Fiedler, F. Haft, R. Traunmüller (eds.): *Expert systems in law. Impacts on legal theory and computer law.* Attempto Verlag, Tübingen, 1988, 47–65.