# On the relation between legal language and legal argument: assumptions, applicability and dynamic priorities

Henry Prakken*
Computer/Law Institute, Free University
De Boelelaan 1105 Amsterdam
email: henry@rechten.vu.nl

Giovanni Sartor
CIRFID, University of Bologna
Via Galliera 3, 40121, Bologna
IDG-CNR, Via Panchiatichi 56/16, Firenze
email sartor@cirfid.unibo.it

## Abstract

This paper investigates how the logical aspects of legal language determine possibilities to attack and defend arguments in a dispute. An argumentation framework in logic-programming style is presented which can cope with the interplay between assumptions, applicability statements and reasoning about preference relations between norms. The theory is given a solid semantic foundation, and can be used to formalise a wide range of realistic legal examples.

## 1 Introduction

In recent years, adversarial legal argumentation has been a major topic of AI & Law research, approached from different angles. Some researchers have developed descriptive theories of legal argument, or even programs that generate disputes (e.g. the work of Ashley, Rissland and Skalak; for an overview see [13]). Others, including the present authors, have studied the logical notions involved in legal argumentation, and again others, e.g. [5] have formalised procedural aspects of legal argument. In our view, these approaches complement each other: descriptive theories and argument generating programs need formal and procedural standards for correct, or good, argumentation, and procedural models of dispute are built around a logical theory of notions like 'argument', 'counterargument' and 'defeat'. For instance, Gordon's Pleadings game is based on [4]'s analysis of these notions. Finally, a logical theory of argumentation is sterile if it is not embedded in actual reasoning practice.

The present paper extends the logical line of research. Our aim is to develop a formal theory of adversarial argument that takes as many 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. Several researchers have addressed some of these features (see section 6 for an overview), but to our knowledge no one has yet developed a logical analysis of the combination of all of them.

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, to make the discussion of the other types of complexity more transparent, but also to reduce the computational complexity and increase

*Henry Prakken was supported by a research fellowship of the Royal Netherlands Academy of Arts and Sciences, and by Esprit WG 8319 'Modelage'. The authors thank Gerhard Brewka and Mark Ryan for some valuable technical suggestions.

0

the prospects for logic-programming implementations.

This paper is also relevant for theories about nonadversarial aspects of legal reasoning. The structures we study are often used deliberately 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.

The theory presented in this paper is an extension of our own previous work, reported in e.g. [10, 11, 14, 15]. What is added is the possibility to express assumptions by means of a weak negation operator, and a way of reasoning *about* priorities, i.e. the possibility to derive the ordering on rules from the rules themselves. Moreover, we state our theory in terms of a fixpoint semantics, and prove that it is well defined for every input theory (we here take 'semantics' to mean a specification of what a program should compute).

## 2 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.

When arguments use rules with contradictory consequents, we will say that these arguments rebut each other. In statutory texts rules with contradictory consequents can be found for a variety of reasons, extensively discussed in our earlier work. In legal debates rules with contradictory consequents typically occur 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.

We now give one example of rebutting arguments, with 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 the conflict between arguments with incompatible rules priority rules are needed, establishing which one of the competing rules is to prevail. The law offers well-known abstract priority rules such as the principles of speciality (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 specific priority assertions, which the interpreter may detect, just by a combined interpretation of the involved rules. 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, than the communication rule should be preferred.

It is important to note that priority rules behave exactly like any other legal rule: they make their consequent (a priority assertion) dependent on the satisfaction of certain conditions, and also their conflicts have to 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 5, 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. 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, for instance, in the above pasta example.

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.

# 3 The formal system

Our system contains the following elements. As preliminaries it has an underlying formal language and, based on the language, a notion of an argument. Its adversarial aspects are a definition of when an argument is in conflict with, or attacked by other arguments, a way of comparing an argument with each individual attacker and, most importantly, a definition of the ultimate status of an argument, in terms of three classes: arguments with which a dispute can be 'won', respectively, 'lost' and arguments which leave the dispute undecided.

## 3.1 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. The language has two kinds of negation, weak and classical negation. An atomic first-order formula is a *positive* literal; a positive literal preceded by $\neg$ is a *negative* literal; a positive or negative literal is a *strong* literal and a strong literal preceded by $\sim$ is a *weak* literal. For any atom $P(x)$ we say that $P(x)$ and $\neg P(x)$ are the complement of each other; in the metalanguage $COMP(L)$ denotes the complement of $L$.

Now a *rule* is an expression of the form

$$r : L_1 \wedge \ldots \wedge L_j \wedge \sim L_k \wedge \ldots \wedge \sim L_m \Rightarrow L_n$$

where $r$ is the name of the rule and each $L_i$ 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.

Finally, the input information of our system is called an *ordered theory*, which is a pair $(T, <)$, where $T$, the set of *premises*, is a set of rules, and $<$ is a noncircular ordering on $T$.

## 3.2 Arguments

The basic notion of our system is that of an argument, which we define in a way that comes close to Dung's [2] notion of a 'defeasible proof'. We regard an argument as a minimal sequence of rules that can be chained together. More precisely, an *argument* is a finite nonempty sequence $r_1, \ldots, r_n$ of ground instances of rules such that, firstly, for every $i, 0 < i \leq n$, for every positive or negative literal $L$ in the antecedent of $r_i$ there is a $j < i$ such that $L$ is the consequent of $r_j$; and secondly, for no $r_i$ its consequent is also the consequent of some $r_j$ ($j < i$). An argument $A$ is *based on* the ordered theory $(T, <)$ iff all rules of $A$ are ground instances of rules in $T$. Where there is no danger of confusion, we will leave the phrase 'based on $(T, <)$' implicit.

We will also use the following notions. An argument $A'$ is a *(strict) subargument* of an argument $A$ iff $A'$ is a (strict) subsequence of $A$. Furthermore, for any argument $A$ the *top rule* of $A$, denoted by $top(A)$, is the last rule of $A$; the *conclusion* of $A$, denoted by $CONC(A)$, is the consequent of $top(A)$; $SUB(A)$ is the set of all subarguments of $A$; and finally, the set of *assumptions* of $A$, denoted by $ASS(A)$ is the set of all literals $L$ such that $\sim COMP(L)$ occurs in $top(A)$.

The following example illustrates these notions: for the argument

$A = (r_1\colon \sim a \Rightarrow b,\ r_2\colon b \wedge \sim d \Rightarrow c)$

- $CONC(A) = c$;
- $SUB(A) = \{(r_1), (r_1, r_2)\}$;
- $top(A) = r_2$;
- $ASS(A) = \{\neg d\}$.

### 3.3 Relations between arguments

So far the notions have been fairly standard; now we will present the features that capture the adversarial aspects of argumentation. We start with defining how the ordering on arguments is determined by the ordering on rules. The idea is that arguments are compared with respect to the last rule they are using. Note, however, that for the general theory this particular method is not essential; other options are possible but because of space limitations we will not discuss them here.

**Definition 3.1 (argument orderings)** *For any ordered theory $\Gamma = (T, <)$ the ordering relation $\ll$ on arguments is defined as follows: $A_1 \ll A_2$ iff $top(A_1) < top(A_2)$.*

Next we define in which ways an argument can attack 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. In section 2 we saw two kinds of conflict between legal rules: rules with contradictory conclusions and rules denying an assumption of another rule. Accordingly, we define two ways of attacking an argument, depending on the difference between weak and classical negation. The first way, rebutting, is when the conclusion of one argument is the complement of the conclusion of another argument. Clearly, this way of attack is symmetric. The second way of attacking an argument, undercutting it, is by attacking an assumption of an argument; this is when the conclusion of an argument is the complement of that assumption. Clearly, this form of attack is not symmetric.

**Definition 3.2 (conflicts between arguments)**

1. *An argument $A_1$* attacks *(or: is a counter-argument of) an argument $A_2$ iff $A_1$ rebuts or undercuts $A_2$.*
2. *$A_1$* rebuts *$A_2$ iff $CONC(A_1)$ and $CONC(A_2)$ are complementary;*
3. *$A_1$* undercuts *$A_2$ iff $CONC(A_1)$ is the complement of some assumption of $A_2$.*

Next we define when an argument is coherent.

**Definition 3.3 (coherency of arguments)** *An argument $A$ is* coherent *iff none of $A$'s subarguments attacks one of $A$'s subarguments.*

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

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 a binary relation of defeat among arguments. 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. Note also that the argument relations and properties defined below are relative to a given ordered theory.

Our definition of 'defeat' is based on two ideas. The first idea is that, in order to defeat an argument, a counterargument can point its attack at the argument itself, but also at one of its proper subarguments, since an argument cannot be stronger than its weakest link. The second idea is that for defeating an argument by undercutting it, no priorities are needed; they are only needed if an argument is rebutted. This agrees with reading $\sim A$ simply as the assumption that there is no evidence of $A$.

**Definition 3.4 (argument relations)** *Let $A_1$ and $A_2$ be two arguments. Then*

1. $A_1$ defeats $A_2$ *iff*

   - $A_1$ *undercuts some subargument* $A_{2'}$ *of* $A_2$*;*

   *or else*

   - $A_1$ *rebuts some subargument* $A_{2'}$ *of* $A_2$ *and* $A_1 \not\leq A_{2'}$*, and* $A_2$ *does not undercut some subargument of* $A_1$*.*

2. $A_1$ properly defeats $A_2$ *iff* $A_1$ *defeats* $A_2$ *and* $A_2$ *does not defeat* $A_1$*.*

3. $A_1$ questions $A_2$ *iff*

   (a) $A_1$ *is coherent, and*

   (b) *For some subargument* $A_{2'}$ *of* $A_2$*:*

      i. $A_1$ *attacks* $A_{2'}$*; and*

      ii. $A_{2'}$ *does not properly defeat* $A_1$*.*

### 3.4 The status of arguments

The central part of our system is a definition of the status of arguments. To prevent circularities it states conditions on *sets* of arguments. Note that also this definition is relative to a given ordered theory. It 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.

First the definition tells us which arguments are winning, or 'justified'. Obviously, such arguments must be coherent but they must also properly defeat all those counterarguments that are not properly defeated by other justified arguments; the 'that' clause allows for arguments reinstating other arguments that cannot defend themselves. Once we know which arguments are justified, we can define the losing, or 'overruled' arguments as those that are attacked by a justified argument and, finally, the undeciding or 'defensible' arguments as all the arguments that are neither justified nor overruled.

**Definition 3.5 (assessment of arguments)** *For any argument A:*

1. *A is* justified *iff A is in the smallest set JustArgs of arguments satisfying the following conditions:*

   (a) *all members of JustArgs are coherent, and*

   (b) *all arguments questioning a member of JustArgs are properly defeated by a member of JustArgs.*

2. *A is* overruled *iff*

   (a) *A is incoherent, or*

   (b) *A is properly defeated by a justified argument.*

   (c) *A is* defensible *iff it is neither justified nor overruled.*

It can be shown that this definition satisfies the 'weakest link' principle.

**Proposition 3.6** *If an argument is justified, all its subarguments are justified.*

*Proof:* This follows from the fact that an argument questioning a subargument of a justified argument also questions the justified argument itself, and is therefore by definition 3.5 properly defeated by a justified argument.□

It is also easy to see that, because of the definition of proper defeat, the set JustArgs contains no arguments attacking each other.

An alternative characterisation of the set of justified arguments is given by the following fixpoint definition, which is similar to [2]'s grounded (sceptical) semantics of logic programs. It is this definition, and its extended version in section 4, that guarantees the existence of a unique set of justified arguments, and thus gives our system a well- defined semantics.

**Definition 3.7** *Let for any ordered theory* $\Gamma$ *Args be the set of all arguments on the basis of* $\Gamma$. *Then we define for any* $\Gamma$ *a sequence of sets of arguments as follows.*

- $JustArgs_0$ *is the set of all arguments in Args that properly defeat all their counterarguments in Args;*
- $JustArgs_{i+1} = JustArgs_i \cup \{A \in Args \mid A$ *properly defeats all its counterarguments in Args that are not properly defeated by members of* $JustArgs_i\}$.

**Proposition 3.8** *Every ordered theory has a unique set of justified arguments.*

*Proof:* It is easy to see that for any $\Gamma$ the definition of the sequence of sets $JustArgs_i$ is equivalent to the iterated application to the empty set of a monotonic operator $F_\Gamma : Pow(Args) \longrightarrow Pow(Args)$. Then $\cup_{i=0}^{\infty}(JustArgs_i)$ is equal to the least fixpoint of $F_\Gamma$. Clearly this set satisfies definition 3.5 and is the smallest set doing so.□

### 3.5 Illustration of the definitions

The first example illustrates the distinction between weak and strong negation: rule $r_1$ states that, if it cannot be shown that a person is a minor, s/he 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 exercize 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'.

$$\begin{array}{ll} r_1(x): & \sim x \text{ is a minor} \Rightarrow \\ & x \text{ has legal capacity} \\ r_2(x): & \neg\, x \text{ is a minor} \Rightarrow \\ & x \text{ has the right to vote} \end{array}$$

The following example illustrates the absolute preference of undercutting over rebutting attack.

$$\begin{array}{llll} r_1: & \Rightarrow a & r_2: & a \Rightarrow b \\ r_3: & \sim b \Rightarrow \neg a & & \end{array}$$

Assume that $r_1 < r_3$. Definition 3.4 ignores this ordering; it always prefers undercutting over rebutting arguments: in this example it causes the argument $(r_1, r_2)$ being justified. The reason is that it undercuts $(r_3)$, while, although $(r_3)$ rebuts $(r_1)$ and $r_1 < r_3$, this does not imply that $(r_3)$ defeats $(r_1, r_2)$, since the latter argument undercuts it.

Finally, we illustrate reinstatement and the weakest link by changing $r_3$ into $r_{3'}$: $\sim b \Rightarrow c$ and adding $r_4$: $\Rightarrow \neg a$ with $r_1 < r_4$. Then $(r_4)$ properly defeats $(r_1)$ and thus also $(r_1, r_2)$, which makes both arguments overruled, while with $(r_4)$ also $(r_{3'})$ becomes justified.

## 4 Reasoning about priorities

So far we have simply assumed that there is an input ordering on the rules. Now we will study the situation where the priorities must themselves be derived from the premises. To this end, we assume that our language contains a distinguished twoplace predicate symbol $\prec$ and we define $x \not\prec y$ as $\neg x \prec y$. Next we change the definition of rebutting arguments, by using an unpublished idea of Gerhard Brewka. Although logically it would be easy to prevent this change, by extending our system with first-order formulas and using them for expressing the properties of $\prec$, we have chosen not to do so because of the increase in computational complexity; instead we 'lift' to the metatheory exactly that part of our vocabulary and of first-order reasoning that we need for satisfying noncircularity of $\prec$.

**Definition 4.1** *An argument* $A_1$ rebuts *an argument* $A_2$ *iff*

1. $CONC(A_1)$ *and* $CONC(A_2)$ *are complementary; or*

2. *$CONC(A_2) = x \prec y$ and $CONC(A_1) = z \prec x$ and $CONC(SUB(A_1))$[1] contains a sequence $y \prec z', \ldots, z'' \prec z$; or the same condition holds with $A_1$ and $A_2$ reversed.*

We now define the consequences of the extended framework with a revision of the fixpoint definition. The idea is that now with the set of justified arguments also the ordering is 'constructed' step- by-step. The resulting definition is inspired by unpublished ideas of Gerhard Brewka. Note that it is now necessary to make the dependence of the various notions on an ordered theory explicit, since the ordering component of that theory now varies with the index of the pairs in the sequence.

**Definition 4.2** *For any set $T$ of rules let $Args$ be the set of all arguments containing only rules of $T$. Then we define a sequence of pairs $(<_i, JustArgs_i)$ as follows. $(<_0, JustArgs_0)$ is such that $<_0 = \emptyset$ and $JustArgs_0$ is the set of all arguments in $Args$ that on the basis of $(T, <_0)$ properly defeat all their counterarguments in $Args$. Then $(<_{i+1}, JustArgs_{i+1})$ is such that*

- *$<_{i+1} = <_i \cup \{r < r' \mid$ for some $A \in JustArgs_i : CONC(A) = r \prec r'\}$;*
- *$JustArgs_{i+1} = JustArgs_i \cup \{A \in Args \mid$ on the basis of $(T, <_{i+1})$ $A$ properly defeats all its counterarguments in $Args$ that are not properly defeated on the basis of $(T, <_{i+1})$ by members of $JustArgs_i\}$.*

*Then the set of arguments that are justified on the basis of $T$ is defined as $\cup_{i=0}^{\infty} JustArgs_i$.*

This definition is equivalent to iterated application to $(\emptyset, \emptyset)$ of an operator $G_T : (T \times T) \times Pow(Args) \longrightarrow (T \times T) \times Pow(Args)$. Obviously $G_T$ is monotonic in its first argument. Then it can be shown that the relation of proper defeat on $Args$ grows with the growth of $<$, which makes $G_T$ also monotonic in its second argument, and ensures that it has a least fixpoint. Clearly, if the least fixpoint of $G_T$ is $(<, JustArgs)$, then $JustArgs$ is the smallest set satisfying definition 3.5 for the ordered theory $(T, <)$.

[1] If $S$ is a set of arguments, then $CONC(S)$ is short for the set of conclusions of all members of $S$.

# 5 Applications

Below 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 default.

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,

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

$r_1(r_3(x), r_4(x))$ 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 [16].

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

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

We next formalise the building regulations example, applied to a case with $Villa_0$ (of rules with an empty antecendent we leave the arrow implicit).

| | |
|---|---|
| $a_1(x)$: | $x$ is a protected building $\Rightarrow$ $\neg$ $x$'s exterior may be modified |
| $a_2(x)$: | $x$ needs restructuring $\Rightarrow$ $x$'s exterior may be modified |
| $a_3(x,y)$: | $x$ is a rule about the protection of artistic buildings $\wedge$ $y$ is a town planning rule $\Rightarrow y \prec x$ |
| $a_4(a_1(x))$: | $a_1(x)$ is a rule about the protection of artistic buildings |
| $a_5(a_2(x))$: | $a_2(x)$ is a town planning rule |
| $a_6(Villa_0)$: | $Villa_0$ is a protected building |
| $a_7(Villa_0)$: | $Villa_0$ needs restructuring |

Clearly, there is the justified argument $A_1$ using the proper instances of[2] $(a_6, a_1)$ for "$\neg$ $Villa_0$'s exterior may be modified", supported by the preference argument $A_2 = (a_4, a_5, a_3)$.

Let us now assume that our rule set includes the temporality principle $T(x,y)$ and that:

$a_8(x,y)$: $a_1(x)$ is earlier than $a_2(y)$

Then we have two merely defensible arguments, $A_1$ and $A_3 = (a_7, a_2)$, which is supported by the preference argument $A_4 = (a_8, T)$.

Let us finally assume that the temporality rule itself also has a temporal dimension, and in particular that:

$a_8(a_3(x,y), T(w,z))$: $T(w,z)$ is earlier than $a_3(x,y)$

By applying the temporality rule to the relation between this rule itself and rule $a_3(x,y)$ we conclude that $T(a_2(Villa_0), a_1(Villa_0)) \prec a_3(a_1(Villa_0), a_2(Villa_0))$. This makes the argument for the prevailence of rules concerning the protection of artistic buildings justified, which in turn makes the argument for the prohibition of modifying the external form of $Villa_0$ justified. Note, finally, that, although the $T$ rule is applied to itself, there is no logical self reference, since one instance of that rule refers to a different instance of the same rule.

Consider now the pasta example, which combines priorities and assumptions.

[2] To maintain readability we only give the function symbol part of the rule names.

| | |
|---|---|
| $b_1(x,y,z)$: | $x$ can be sold in $y$ $\wedge$ $y$ is an E.C. country $\wedge$ $z$ is an E.C. country $\wedge$ $\sim$ the selling of $x$ in $z$ endangers public health $\wedge$ $\sim$ the selling of $x$ in $z$ prejudices the consumer $\Rightarrow$ $x$ can be sold in $z$ |
| $b_2(x, Italy)$: | $x$ is called "pasta" $\wedge$ $\neg x$ is made of hard corn $\Rightarrow$ $\neg x$ can be sold in $Italy$ |

Let us assume the following facts

| | |
|---|---|
| $b_3(BRD)$: | $BRD$ is an EC-country |
| $b_4(Italy)$: | $Italy$ is an EC-country |
| $b_5(P_0, BRD)$: | $P_0$ can be sold in $BRD$ |
| $b_6(P_0)$: | $P_0$ is called "pasta" |
| $b_7(P_0)$: | $\neg P_0$ 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 $B_1 = (b_3, b_4, b_5, b_1)$, which defeats the rebutting argument $B_2 = (b_6, b_7, b_2)$. However, if we also ascertain that:

$b_8(P_0, Italy)$: the selling of $P_0$ in $Italy$ prejudices the consumer

then argument $B_1$ is undercut by argument $B_3 = (b_8)$, so that the before overruled argument $B_2$ for "$\neg P_0$ can be sold in $Italy$" becomes justified.

Finally, we show how statements about applicability can be formalised. In fact, such statements correspond to Pollock's [9] 'undercutting defeaters'. However, unlike Pollock we can deal with the interaction between undercutting defeaters and priorities, and with debates about applicability. There are two alternatives, both using a special applicability predicate *appl*. The first method is based on the view that legally any rule can make any other rule applicable, regardless of their priority. This is easy to formalise, by simply giving each rule $r$ an extra condition $\sim \neg appl(r)$. Then any argument with as conclusion $\neg appl(r)$ undercuts an argument with $r$.

The alternative method can be used if one thinks that a legal rule can only make another rule inapplicable if it is not of lower priority; it was earlier discussed in [10, 11] but now it can be combined with the other features of our framework. The method consists of adding to every rule $r$ in $T$ a condition $appl(r)$ and, moreover, to add a new rule $r_0$ to $R$ of the form $r_0 : appl(r)$ (note that $r$ here is a variable). The idea is that any instance of $r_0$ with name $n$ receives the same Lex Superior and Lex Posterior relations as the rule named $n$; then arguments with top rule $n' : L \Rightarrow \neg appl(n)$ rebut and defeat arguments using $n$, because of Lex Specialis.

# 6 Related work

Both inside and outside logic programming rule systems with a syntax similar to ours have been investigated and most of these systems contain elements of the present system. However, to our knowledge, the present framework is the first that both studies the combined treatment of the various features and provides a well- defined semantics by proving both uniqueness, coherence and guarantee of existence of the set of justified arguments.

Because of space limitations we confine ourselves to some brief remarks. In logic programming, Dung [2] has studied the combination of rebutting and undercutting arguments but without priorities, while 'ordered logic' (see e.g. [3]) does have priorities but no undercutting arguments or dynamic priorities. Nute's [8] 'defeasible logic' also has ordered one- direction rules combining literals. The logic has static but no dynamic priorities and does not allow for undercutting arguments. Finally, the system exhibits an undesirable, extreme form of scepticism (cf. [10, pp. 158-60]). Another related system is Brewka's [1] prioritised version of normal default logic, which has dynamic priorities but no weak negation, nor a notion of an argument. A problem is that existence of extensions is not guaranteed. Our treatment of reasoning about priorities was inspired by unpublished attempts of Brewka to solve this problem.

Also some projects in AI and Law should be mentioned. Two logic programming projects are [17] and [7]. In [17] the priority rules are not represented in the knowledge representation language but encoded in the inference engine. [7] use the DEMO predicate of logic metaprogrmamming instead of priorities, which makes their style of representation very different from ours. Moreover, they assume a hierarchy of separated levels of legal knowledge. We have chosen not to make this assumption, because of examples like our building regulations, where Lex Posterior not only applies to object level rules, but also to conflict resolution rules, including itself.

Like the present paper, also Hage (e.g. [6] has stressed the need for combining reasoning with and about priorities and about applicability; in fact, he reduces priority statements to applicability statements. His system, which is meant to be a formalisation of Raz' [12] theory of reasons, does not have the notion of an argument.

Finally, [5], which contains many interesting examples and observations, has all the elements of the present framework. However, there are two important differences. Firstly, Gordon's account of reasoning about priorities is not formulated as a logic but as a method of representing knowledge in a particular logic, [4]'s conditional entailment. And secondly, his use of [4] makes specificity the overriding source of priorities.

# 7 Conclusion

Inspired by the richness of legal language, this paper has presented a semantic foundation for the combination of several features of legal argument that were previously mainly studied in isolation: reasoning with priorities, reasoning about priorities and reasoning with assumptions and applicability statements. At the technical level it is ensured that every input theory has precisely one set of justified arguments, which, moreover, is conflict free. As for applications we have shown that the system can be used to formalise a wide range of realistic legal exampes. On some points alternative definitions may be possible: for example, with respect to the input ordering on arguments and the relation between undercutting and rebutting defeat. Yet we think that it is a merit of our paper that by providing a formalisation it has raised these issues.

One aim of future research is to extend the expressiveness of the language in computationally feasible ways. In particular, we want to have ways of expressing properties of relations, such as transitivity of relations like $\prec$, 'inferior' and 'earlier', and of expressing the incompatibility of predicates like 'bachelor' and 'married'. We also plan to study implementation, along two lines: firstly, developing a logic programming metainterpreter in the style of [14, 15] for two predecessors of the present framework; and, secondly, adapting proof theoretical ideas of [3, 4, 8] in terms of 'dialectical trees'.

## References


[1] G. Brewka, Reasoning about priorities in default logic. *Proceedings AAAI-94.*

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

[3] D. Gabbay, E. Laenens and D. Vermeir, Credulous vs. sceptical semantics for ordered logic programs. *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann 1991, 208-217.

[4] H. Geffner and J. Pearl, Conditional entailment: bridging two approaches to default reasoning. *Artificial Intelligence* 53 (1992), 209-244.

[5] T.F. Gordon, The pleadings game - formalizing procedural justice. *Proceedings of the fourth International Conference on Artificial Intelligence and Law*, Amsterdam 1993, ACM Press, 1993, 10-19.

[6] J.C. Hage and B. Verheij, Reason-Based Logic: a logic for reasoning with rules and reasons. To appear in *Law, Computers and Artificial Intelligence.*

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

[8] D. Nute, Defeasible logic. In D. Gabbay (ed.) *Handbook of Logic and Artificial Intelligence*, Oxford University Press, 1993, 353-395.

[9] J.L. Pollock, Defeasible reasoning. *Cognitive Science* 11 (1987), 481-518.

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

[11] H. Prakken, An argumentation framework in default logic. *Annals of Mathematics and Artificial Intelligence*, 9 (1993), 93-132.

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

[13] E.L. Rissland, Artificial Intelligence and Law: stepping stones to a model of legal reasoning. *Yale Law Review* vol. 99 (1990), 1957-1981.

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

[15] G. Sartor, A formal model of legal argumentation. *Ratio Juris* 7 (1994), 212-226.

[16] G.R. Simari and R.P. Loui, A mathematical treatment of defeasible argumentation and its implementation. *Artificial Intelligence* 53 (1992), 125-157.

[17] H. Yoshino and M. Kithahara, 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.