A tool in modelling disagreement in law: preferring the most specific argument

Henry Prakken
Computer/Law Institute
Vrije Universiteit Amsterdam

Abstract

This paper presents a formal theory about preferring the most specific argument. The theory is applied to legal reasoning and used to formulate requirements for legal knowledge-based systems choosing between alternative arguments. It is based on a proposal of Poole, but improves it in two respects: firstly, default logic is shown to be a better underlying logic for defeasible reasoning than standard first-order logic; and secondly, specificity is defined iteratively, in order to handle multiple conflicts and to characterize the set of preferred knowledge. The theory is an example of the fact that logic can be a tool in legal reasoning even if deduction is not regarded as the right way to model it.

1. Introduction

In response to "naive rule-based" developments in the field of AI and law there has been an increasing interest in legal AI systems which can give and compare possibly conflicting alternative solutions to a legal problem. Examples are the Hypo system (Ashley and Rissland, 1987), the system of Gardner (1987) and the Prolexs system (Oskamp et al., 1989). It might be argued that this development implies a shift from logical to other methods in modelling legal reasoning; in this paper, however, I will show that, even if deduction is not regarded as appropriate to model disagreement in law, logic can still be useful as a tool in legal reasoning. I will do so by investigating a logical tool in comparing alternative solutions: preferring the most specific argument.

In various respects a study of the specificity principle can contribute to the field of AI and Law. Firstly, this principle is, at least for continental systems, generally accepted as legally valid for regulational sources of legal knowledge, and therefore lawyers are expected to prefer the most specific regulation; as a consequence, in regarding cases with alternative solutions as easy if one of them is based on a more specific argument, the principle draws part of the boundary between "hard" and "easy" questions, which is relevant for systems for "issue spotting" (Gardner, 1987; Gordon, 1989). Secondly, in solving legal problems it is often necessary to assume of a case that it is normal if nothing is known about the existence of exceptions (cf. Gardner, 1987:55-9); using the specificity-principle is a way to make this assumption.

This last aspect makes the present study also relevant for the general AI-study of so called "nonmonotonic" or "common-sense" reasoning, which is a kind of reasoning in which conclusions may become invalid if further information is given. Although formalisms for nonmonotonic reasoning are generally motivated by referring to the problem of handling exceptions, only some such systems actually incorporate the principle that exceptions defeat general rules. An example is the system of Delgrande (1987), in which the principle is incorporated into a possible-worlds approach to defeasible reasoning. Another approach is to use some kind of consistency- or nonprovability-operator in combination with exception clauses, either formulated specifically, as e.g. in Etherington and Reiter (1983) and McCarty (1988), or generally, as e.g. in Routen (1989), containing a legal implementation in PROLOG.

However, for philosophical reasons I will concentrate on using the principle as a metarule for choosing between competing arguments, since as such it more naturally fits into the "modelling disagreement" view on legal reasoning than the other approaches, which do not allow competing arguments. Examples of this approach in the general AI-literature are Poole (1985) and Loui (1987). The aim of this paper is to develop a formal theory about preferring the most specific argument: rather than giving a procedure to determine which arguments are preferred, the theory will give definitions of what it means if an argument is preferred; thus it can be used as a touchstone for implementations of "specific defeats general".

My investigations will be formal in nature; the reader is assumed to be familiar with first-order predicate logic and not totally unfamiliar with the study of nonmonotonic reasoning. The starting point of my research is the approach of Poole (1985). In section 2, following an overview of his ideas, some problems are identified which motivate an improved and extended definition of the specificity principle, given in secton 3, and applied to some examples in section 4; section 5 is about implementation.

2. Poole: preferring the most specific explanation

2.1. Poole's theory comparator

Poole (1985) presents a formalization of the "Specific defeats general" principle against the background of a general view on default reasoning presented in detail in Poole (1988). Essentially, this view is that if defaults are regarded as possible hypotheses with which theories can be constructed to explain certain facts, there is no need to change the logic but only the way the logic is used. Accordingly, the semantics and proof theory of Poole's "logical framework for default reasoning" are simply those of first order predicate logic. The basis of this framework are the sets F and δ. F is a set of closed first-order formulas, the facts, assumed consistent; and δ is a set of possibly inconsistent first-order formulas, the defaults or possible hypotheses. A scenario of a pair (F,δ) is a consistent set F U D, where D is a set of ground instances of defaults of δ. An explanation of a closed formula is a scenario implying it. Theory formation consists of constructing an explanation for a given formula. These definitions say that in constructing an explanation the facts must be obeyed but that the use of any default is free, as long as, when taken together, they are consistent with the facts. Conflicting explanations can be compared with respect to any criterion, one of which is specificity.

What is striking in Poole's view on default reasoning is its similarity to the "modelling disagreement" view on legal reasoning (cf. Gordon, 1989); the legal counterpart of explanations are arguments for a desired solution of a case: certain facts must be obeyed by such arguments: for example, facts about the case at hand, or necessary truths such as "a man is a person" or "a rent contract is a contract", but for the rest a lawyer has available a large body of conflicting opinions, rules, precedents etc.. from which to choose a coherent set of premises which best serve the client's interests. Also viewing "specific defeats general" as a choice between competing explanations nicely fits into the "modelling disagreement" view on legal reasoning.

Although Poole (1985) is mainly concerned with inheritance networks, he does not restrict his specificity principle to such networks, but defines it on the semantics of full first-order predicate logic. Consider explanations Ai = F U Di for α and Aj = F U Dj for β (Greek letters α,β and τ, as well as letters a,b,c, etc., are used in this paper as metavariables for arbitrary first-order formulas). The facts of Ai and Aj can be divided into necessary facts Fn, true in all explanations based on (F,δ), and contingent facts Fc, the "input facts". In determining specificity only the necessary facts are taken into account. The reason will be explained below in the discussion of the "loose bricks" example. Now Ai is <u>more specific than</u> Aj iff there is a possible fact Fp which makes Aj explain β without making Ai explain α or β (without this last requirement for Ai and β, Fp = β would always make Ai more specific than Aj if $\alpha \neq \beta$). In formal notation, iff

Aj = {Fp} U Fn U Dj $\models$ β
Ai = {Fp} U Fn U Di not $\models$ α and not $\models$ β.
($\models$ denotes first-order entailment).
If, in addition, Aj is not more specific than Ai, Ai is <u>strictly more specific than</u> Aj.

A few examples illustrate the definitions (which are slightly different than those of Poole). Consider first a pair of rules stating that anyone who has borrowed money must pay it back, unless another person has payed it back for him or her. Let us assume that this is the case with Bob, who has borrowed 50 pounds: in predicate logic this may be formalized as

1. Borrowed(Bob,£ 50) -> Must_pay_back(Bob,£ 50)
2. [(Borrowed(Bob,£ 50) & Payed_by_third(£ 50)]
   -> ¬Must_pay_back(Bob,£ 50)]

Fc = {Borrowed(bob,£ 50), Payed_by_third(£ 50)}
δ = {1',2'} where 1',2' are 1,2 with the constants replaced by variables. In the following examples δ will be left implicit.

A1 = Fc U {1} is an argument for Must_pay_back(Bob,£ 50), while A2 = Fc U {2} is an argument for the opposite. A2 defeats A1 since the antecedent of (2) logically implies the antecedent of (1) while the reverse does not hold; this means that, on the one hand, every fact which makes A2 explain ¬Must_pay_back(Bob,£ 50) makes A1 explain the opposite while, on the other hand, there is a fact, Borrowed(Bob,£ 50), which makes A1 explain Must_pay_back(Bob,£ 50) without making A2 explain its negation. Therefore, the argument for ¬Must_pay_back(Bob,£ 50) takes precedence.

Another typical case of specificity occurs when one antecedent implies another merely as a matter of fact. Consider the example of a rule stating that contracts bind only the parties involved, and another rule saying that rent contracts of houses also bind new owners of the house. For a given contract c this is formalized as

3. Contract(c) -> Binds_only_parties(c)
4. HouserentContract(c) ->
   (¬Binds_only_parties(c) & Binds_all_owners(c))

Fn = {(x)[HouserentContract(x) -> Contract(x)]}
Fc = {HouserentContract(c)}

The argument A3 = Fn U Fc U {3} explains Binds_only_parties(c), while A4 = Fn U Fc U {4} explains ¬Binds_only_parties(c) & Binds_all_owners(c). A4 is strictly more specific than A3: Contract(c) is a possible fact which makes A3 explain Binds_only_parties(c) without making A4 explain (¬Binds_only_parties(c) & Binds_all_owners(c)) or Binds_only_parties(c), and therefore A4 is more specific than A3; on the other hand, A3 is not more specific than A4, because every fact which implies HouserentContract(c) and thus makes A4 apply, because of Fn also implies Contract(c), which makes A3 apply.

A more complicated type of examples is of the following logical form:

D5 = { a -> b, b -> c }
D6 = { (a & e) -> d, d -> ¬c }
Fc = {a,e}

According to Poole the explanations A5 = D5 U Fc for c and A6 = D6 U Fc for ¬c are both more specific than each other: b is a possible fact which makes A5 applicable and not A6, and d is a fact which makes A6 applicable and not A5 (recall that in determining specificity Fc is ignored). At first sight, however, it seems that there is a reason to prefer A6, viz. the fact that it is based on the fact situation (a & e), which is a specific instance of the fact situation a on which A5 is based. Loui (1987), calling this a case of "superior evidence", does indeed define "specific defeats general" in such a way that it prefers A6. A legal example:

5. If a wall has loose bricks, there is a maintenance deficiency.
6. If a wall near a road has loose bricks, there is a dangerous situation.
7. In case of a maintenance deficiency the landlord, not the tenant, must act.
8. In case of a dangerous situation the tenant, not the landlord, must act.

Fc = A wall has loose bricks and is near a road.

Whereas Loui's definitions prefer the explanation Fc U {6,8} for "the tenant must act", according to Poole's definition the case is ambiguous, which does indeed seem to be the best solution, for the following reasons. In preferring the most specific argument two phases can be distinguished: firstly, determining which argument is the most specific; and secondly, deriving new facts with the preferred argument. In my view Fc only plays a role in the second phase, in determining what may be held on the basis of the facts of the case at hand. On the other hand, specificity is determined with respect to all possible situations; for an argument to be preferred it is not enough to be more specific only under the contingent facts of the case at hand. It is the latter situation which occurs in the "loose bricks" example: the norm (8) itself is, witness its formulation, not meant for a specific kind of maintenance deficiencies but for dangerous situations in general, irrespective of whether they are maintenance deficiencies; therefore in other situations the competing arguments could be ambiguous and, as a consequence, it cannot be said that the normgiver has meant (8) as an exception to (7).

2.2. Problems

Despite their intuitive attractiveness, Poole's ideas do not always give satisfactory results: firstly, his definition of specificity ignores the possibility of multiple conflicts; and secondly, the fact that in his framework for default reasoning defaults are represented in standard logic gives rise to arguments which should not be possible.

a. Multiple conflicts ignored

Poole's definition of specificity handles examples in which more than one conflict must be solved incorrectly, because it ignores the possibility that an argument contains a defeated premise. Consider the following example:

$$D1 = \{ a \rightarrow b, \quad b \rightarrow c, \quad c \rightarrow d \}$$
$$D2 = \{ (a \,\&\, c) \rightarrow \neg b, \quad \neg b \rightarrow e, \quad e \rightarrow \neg d \}$$
$$Fc = \{a,c\} \qquad Fn = \{c \rightarrow e\}$$

Poole's definition prefers A1 = Fc U Fn U D1 for d, because e is a fact which makes A2 = Fc U Fn U D2 explain ¬d without A1 explaining d, while all facts which make A1 explain d imply c and therefore, since (c -> e) is in Fn, also e, which makes A2 explain ¬d (note again that Fc is ignored). However, A1 uses the fact b, for which the explanation A1' = Fc U Fn U {a -> b} is clearly defeated by A2' = Fc U Fn U {(a & c) -> ¬b} for ¬b. Of course, as Poole (1985:146) himself recognizes, for an argument to be preferred not only the final conclusion but also all intermediate conclusions must be preferred.

What is needed is an iterative definition of "specific defeats general": it should be the case that not only the "final" conclusions of an argument are preferred, but also all intermediate conclusions. This means that a fact can only be regarded as preferred if there is a scenario such that <u>every</u> fact that is implied by it has a preferred argument.

b. Defaults cannot be formulas of standard logic

Poole (1988) claims that if his framework for default reasoning is adopted, there is no need to change the logic for defaults, i.e. rules which are subject to exceptions, since they can be simply represented as ordinary first-order formulas. However, if his framework is combined with the view that preferring exceptions is <u>choosing</u> between arguments, there are strong objections to this claim, since using the material implication for defaults makes possible arguments which intuitively should not be possible at all. Consider first the example of Bob having killed Karate Kid in self-defence.

1. Killed(Bob,KK) -> Guilty(Bob)
2. [Killed(Bob,KK) & Self-defence(Bob)] -> ¬Guilty(Bob)
3. Defended_against(KK,Bob) -> Self-defence(Bob)

Fc = {Killed(Bob,KK), Defended_against(KK,Bob)}

Intuitively, the preferred conclusion in this example is with no doubt ¬Guilty(Bob). However, Poole's definition allows us to explain Self-defence(Bob) from Fc U {3}, but also ¬Self-defence(Bob) from Fc U {1,2}; and this would mean that given the premises there is an irresolvable legal issue concerning Self-defence, for which reason the argument for ¬Guilty(Bob) uses a non-preferred subargument and cannot be preferred. However, in legal reasoning arguments like the one for ¬Self-defence(Bob) are not constructed; only arguments for facts which are the consequent of a legal rule are regarded as possible: if legal rules are viewed as defaults they have directionality and therefore Modus Tollens, on which the argument for ¬Self-defence(Bob) is based, should be impossible. Even as an explanation of a decision with hindsight Modus Tollens cannot be used: assume Bob was found guilty, then it is not the case that it must have been found that Bob was not acting in self-defence, since maybe he was, but he was still found guilty on the basis of a rule defeating (3).

In this view, given the premises the only legal issue is the conflict between Fc U {1} for Guilty(Bob) and Fc U {2,3} for ¬Guilty(Bob), of which the second is clearly preferred. This argument seems to hold for nonlegal defaults as well.

It must be admitted that Poole (1988:137-40), recognizing these arguments as "a possible point of view", presents a method, based on naming defaults, to block Modus Tollens for defaults. This method, however, is optional: the choice whether to use it or not must be made separately for each default; philosophically this is not satisfactory: if Modus Tollens is regarded as invalid for defaults, this should be expressed in their logic.

Furthermore, blocking Modus Tollens does not prevent the following problem, which can occur if the specificity rule is iteratively defined on standard logic. In that case defaults must, since they are implied by any explanation in which they are used, at least be preferred themselves, if the scenario is to be capable of explaining any preferred fact at all. However, if there are conflicting explanations, then defaults used in an explanation cannot be explained preferredly, as the following example shows.

4. a -> b   5. (a & c) -> ¬b   Fc: {a,c}

Clearly, our extended theory comparator should deliver A2 = Fc U {5} as the preferred explanation; however, it does not: A1 = Fc U {4} implies (a & c & b), which is equivalent to the denial of (5): ¬((a & c) -> ¬b). In the approach towards multiple conflicts proposed here not only A2 for ¬b, but also A2 for (a & c) -> ¬b should be strictly more specific, because (5) is implied by A2. Unfortunately, however, it is not: there is no fact which makes A1 explain ¬((a & c) -> ¬b) without making A2 explain the unnegated implication, for the latter, being a default, needs no facts at all to be explained.

What causes the problem is the fact that in exceptional cases the general default can be used to set up an argument against the default which is an exception to it: in our example the possibility to explain b with (a -> b) under the circumstance (a & c) is seen as an argument against (a & c) -> ¬b. However, intuitively this is very strange, because it is part of the very meaning of defaults that they can have exceptions: therefore, it should be impossible to use defaults as an argument against exceptions to them. However, if defaults are formalized as material implications, there is no natural way to achieve this.

In conclusion, then, these examples show that Poole's framework for default reasoning cannot be combined with the view that exceptions create alternative arguments. Therefore, something has to be changed. Rather than adopting an approach in which exceptions <u>block</u> more general arguments, for instance, by making them inconsistent, which is one of the ideas behind naming defaults in Poole (1988), I will, as a solution, change his framework in such a way that the idea of specificity as <u>choosing between</u> arguments is retained. Furthermore, in my view specificity should be encoded: the possibility that specificity is determined by some externally defined ordering should not be left open, as e.g. in Brewka (1989), but specificity should, as in Poole (1985), be determined solely by the semantics of the formulas involved in the argument.

3. Specificity defined on default logic

3.1. Default logic

In the remainder of this paper Reiter's default logic (Reiter, 1980) will be used as the underlying logic for the specificity principle. Poole's and Reiter's systems are very similar. Both are based on a set of facts and a set of defaults, and in both systems arguments can be set up by using any default one wishes, as long as consistency is preserved. If as many defaults as possible are thus used, i.e. if adding any new default would cause an inconsistency, sets result which Poole calls maximal scenarios and Reiter extensions. Both can be seen as maximal sets of beliefs which may be held on the basis of certain facts and default assumptions. Since defaults can conflict, there may be more, mutually inconsistent, maximal scenarios or extensions.

A crucial difference between the two systems is that, whereas Poole's defaults are first-order formulas, those of Reiter are inference rules: α:ß/τ informally reads as "If α holds and ß may be consistently assumed, τ may be inferred". α is called the prerequisite, ß the justification, and τ the consequent of the default. It is because of this reading of defaults that defining the specificity criterion on default logic meets two of the requirements formulated in section 2: it is impossible to construct arguments against inference rules; and modus tollens cannot be applied to them.

Another difference is that default logic is nonmonotonic: if a default α:ß/τ is used to infer τ, and after that $\neg$ß is added, then the inference of τ becomes invalid; first-order predicate logic, on the other hand, is monotonic: merely adding premises never invalidates first-order inferences.

3.2. Definitions

Now an extended and improved version of the specificity rule is presented, defined on default logic. Poole's defaults α -> ß will be translated as Reiter's normal defaults α:ß/ß, written as a => b. Normal defaults are defaults of which the justification is identical to the consequent. Observe that using the specificity rule to deal with exceptions is meant to preclude the need for seminormal defaults, i.e. defaults of the form α:(ß & τ)/ß, in which τ is normally a specific exception clause (cf. Touretzky, 1986:20-1). Avoiding such defaults is desirable, because the logic of seminormal default theories is much more problematic than that of normal default theories (Reiter, 1980).

All definitions and proofs below are relative to a fixed default theory (F,δ). The specificity rule is defined on "proof sets", which I define, analogously to a scenario of Poole, as a set of facts and a set of defaults. The idea is that a proof set does not give rise to conflicting beliefs: therefore it should have a unique extension. Furthermore, all defaults should be relevant to the argument: therefore they should be applicable.

Definition 1: a. S = (F,D_i) (where D_i is a finite subset of ground instances of δ) is a proof set (p.s.) iff it has a unique extension E(S) such that of all defaults both the prerequisites and the consequents are in E(S).
b. S explains a formula α iff α is in E(S) or, equivalently (Reiter, 1980:92), iff α is classically implied by the union of F and the set of consequents of all defaults of S.
c. A proof set S' = (F,D') is a sub-proof set of a proof set S = (F,D iff D' C D (C denotes proper inclusion).

A preferred proof set is iteratively defined as follows:

Definition 2: A proof set S = (F,D) is a preferred proof set (p.p.s.) iff

1. All sub-proof sets of S are a preferred proof set;
2. For all α explained by S which are not explained by any sub-proof set of S: if there is a p.s. S' which explains $\neg\alpha$ and which does not interfere with another p.p.s., then S is strictly more specific than S' with respect to α.

Because of the condition that all sub-proof sets of S are also preferred, (1) ensures that multiple conflicts are correctly handled. (2) states that for every fact which is not already explained by a sub-proof set of S, S itself must defeat all arguments for a contradicting fact which do not contain a defeated sub-proof set. Checking whether S succeeds in doing so is the job of Poole's original definition, which is restricted here to a particular formula and adapted to default logic:

Definition 3: S1 = (Fc U Fn,D1) is more specific than S2 = (Fc U Fn,D2) with respect to α (m.s./α) iff: if S2 explains $\neg\alpha$, then there is a possible fact Fp such that:
(Fn U {Fp}),D2) explains $\neg\alpha$;
(Fn U {Fp}),D1) does not explain α;
(Fn U {Fp}),D1) does not explain $\neg\alpha$.
Being strictly more specific is defined as above.

Unlike Poole's definition, the absolute and iterative notion of preferedness provides the opportunity to characterize the set of defeasible knowledge of a default theory, i.e. the facts for which there is an argument which is better than any competing argument; in legal terms: the facts and rules with which a case can be won. Two ways of defining this set suggest themselves; the first is simply to collect all facts which are explained by some preferred argument:

Definition 4: The set of defeasible knowledge $DK_{(F,\delta)}$ of (F,δ) is the set of all formulas explained by a preferred proof set (F,D) such that D $\subseteq$ δ.

In Prakken (1991) a few properties of this DK are discussed: among other things it is shown that DK is closed under first-order logical consequence and that it is the unique extension of (F,D-pref), where D-pref is the set of defaults used in any p.p.s.

Another way to define the preferred knowledge of a default theory is to take the intersection of some of its extensions, viz. of those not containing any defeated formula. To achieve this the specificity-criterion is used to filter the set of extensions (an idea of D.W. Etherington; see Touretzky, 1986:20-1): all defeated extensions are deleted, i.e. extensions which contain the negation of a formula for which there is a preferred proof set.

Definition 5: The set of defeasible knowledge $DK^*_{(F,\delta)}$ of (F,δ) is the intersection of all extensions of (F,δ) which are not defeated.

Proposition 1: DK^* is closed under first-order logical consequence.

Proof: All formulas which are in DK^* are in all extensions of which DK^* is the intersection. Therefore, if a set of formulas of DK^* implies α, this set is in all such E's and, therefore, since extensions are by definition closed under first-order consequence, α is in them as well. By definition α is then in DK^*.■

Proposition 2: If a formula is in DK, it is in DK^*.

Proof: Assume for contradiction that there is a p.p.s. S for α, and DK^* does not contain α. Then some not defeated extension E of (F,δ) does not contain α and therefore, since α is implied by F with the consequents of all the defaults of S (Reiter, 1980:92; cf. definition 1b), this E misses the consequent β of some default of S. Then either β can be consistently added to E, in which case β is in E by definition of an extension (Reiter, 1980:89), whereas by assumption it is not, or it cannot, in which case E contains $\neg\beta$. But then, since β is explained by a p.p.s., E is a defeated extension, which contradicts the observation that it is not.■

The reverse of proposition 2 does not hold; a counterexample is

$$F = \{c,d,e\}$$
$$\delta = \{d \Rightarrow a, \quad (a \,\&\, c) \Rightarrow b, \quad e \Rightarrow \neg a, \quad (\neg a \,\&\, c) \Rightarrow b\}$$

Of the two arguments (F,{d => a}) for a and (F,{e => ¬a}) for ¬a neither is preferred; therefore b, which needs a or ¬a to be explained, is, because of the iterative definition of a p.p.s., not explained by any p.p.s., for which reason it is not in DK. However, b is in DK^*, since this default theory has two extensions:

E1 = Th(F U {a,b})
E2 = Th(F U {¬a,b})

and E1 $\cap$ E2, although it contains neither a, nor $\neg$a, contains b.

Intuitively, the difference between DK and DK* is that DK is a more constructive approach, which only contains facts for which a preferred argument can be constructed, whereas DK* also allows facts to be preferred which hold irrespective of which choice is made in case of conflicting arguments of which neither is strictly more specific.

4. Some applications

In this section the definitions of section 3 are applied to some further examples.

Example 1. The first example shows that the definitions are capable of handling exceptions to exceptions. It is formed by adding to the second example of 2.1. a norm that rent contracts of houses bind all subsequent owners of the house, unless the tenant has agreed by contract with the opposite.

1. Contract(c) => Binds_only_parties(c)
2. HouserentContract(c) => ($\neg$Binds_only_parties(c) & Binds_all_owners(c))
3. (HouserentContract(c) & Tenant_agreed_by(c)) => Binds_only_parties(c)

Fn = {(x)[HouserentContract(x) -> Contract(x)]}
Fc = {HouserentContract(c), Tenant_agreed_by(c)}

S1 = (Fc U Fn,{1}) explains Binds_only_parties(c), S2 = (Fc U Fn,{2}) explains the opposite and S3 again explains Binds_only_parties(c). Clearly S2 is strictly more specific than S1. However, in order to be a p.p.s., S2 must also defeat S3, but it is the other way around, since S3 is strictly more specific than S2. Therefore S3 is a preferred proof set and the consequent of (2) is neither in DK, nor in DK*.

Example 2. This example shows a peculiarity of the second clause of definition 2. To the third example of 2.1. a default is added stating that if a wall near a road which is seldom used has loose bricks, there is no dangerous situation.

1. loose bricks => maintenance deficiency
2. (loose bricks & near road) => danger
3. maintenance deficiency => (landlord & $\neg$tenant)
4. danger => (tenant & $\neg$landlord)
5. (loose bricks & and near road & seldom used) => $\neg$danger

Fc = {loose bricks, near road, seldom used}

Like in 2.1., S1 = (Fc,{1,3}) explains "landlord" while S2 = (Fc,{2,4}) explains "$\neg$landlord". But unlike in 2.1., although not S1 s.m.s./landlord S2, S1 is still a p.p.s., since S2 contains a defeated sub-proof set, (Fc,{2}), which is defeated by (Fc,{5}).

Example 3

D1. Sales-contract(a,b) => Obliged_to_deliver(a) & Obliged_to_pay(b)
D2. Sales_contract(a,b) & Refuses_to_pay(b) => $\neg$Obliged_to_pay(a)

Fc = {Sales-contract(a,b), Refuses_to_pay(b)}

Clearly, $\neg$Obliged_to_deliver(a) is preferred, but what about Obliged_to_pay(b)? There is no proof set for the opposite, but since the only p.s. explaining it is defeated, Obliged_to_pay(b) is neither in DK, nor in DK*. In this respect the present definitions differ from those of Delgrande (1987). In my view, the correct answer in this example depends on whether the two obligations are regarded as connected or not, and since this is not a logical matter, a formal system should have ways to formalize both possibilities. In the present system this can indeed be done: the alternative interpretation can be represented if D1 is split into the next two defaults.

D3: Sales-contract(a,b) => Obliged_to_deliver(a)
D4: Sales-contract(a,b) => Obliged_to_pay(b)

Thus Obliged_to_pay(b) can be explained preferredly with S3 = (Fc,{D4}).

Example 4. This example shows a problem with the definition of a preferred proof set: in some cases it is circular.

D5 = { a => b, (b & c) => $\neg$d }
D6 = { c => d, (a & d) => $\neg$b }
Fc = {a,c}

In order to know whether S5 = (Fc,D5) is a p.p.s. we must first determine whether S5 s.m.s./d S6', where S6' = (F,{c => d}). It appears that this is indeed the case. Furthermore, we must know whether S5' = (F,{a => b}) is a p.p.s.; S6 = Fc U D6 would seem to defeat S5', but actually it does not, since S5 s.m.s./d S6' = (Fc,{c => d}), which is contained in S6. Does this mean that S6 is a defeated proof set? This would be the case if S5 were a p.p.s., but this is what we are trying to find out! Here the definition proves to be circular.

Programmers should be aware of this additional source of circularities (e.g. the program of Nute (1989) does not prevent them). A solution may be to define an ordering which is such that if a default theory satisfies it, circularities will not occur (cf. e.g. Touretzky, 1986).

5. Implementation

As was said in the introduction, the aim of this paper has not been to give a procedure for determining which arguments are preferred, but to give a definition of what it means that an argument is preferred. As a consequence, the theory developed in this paper is not very well suited for a straightforward implementation. Moreover, implementing the full theory is problematic for a number of reasons. Firstly, the theory uses the full expressive power of first-order predicate logic, for which as a whole to date no theorem provers exist which are both complete and efficient. Furthermore, default logic is known to be non-semidecidable: there is no algorithm which garantuees that every provable formula is proven (Reiter, 1980:104). Finally, unlike theorem provers for standard logics, which can stop when a proof has been found, systems which try to find the best argument will have to continue searching the whole space of possible counterarguments.

In practice, problems of efficiency may be overcome by restricting the language, for example to clause logic, as in Nute (1989), or to the even more restricted language of multiple inheritance systems (cf. Touretzky, 1986), of which the expressive powers are, however, too weak for most legal applications. Moreover, efficiency may be increased by sacrificing completeness with respect to our theory.

Nevertheless, however difficult the implementation of the theory developed in this paper may be, it does at least make it possible to formulate exactly in which respects practical applications are or have to be imperfect. Particularly relevant for practical purposes is the following list of requirements which should be met and issues which should be taken into account when implementing "Specific defeats general":

- the program should handle multiple conflicts correctly, i.e. iteratively;
- Modus Tollens may not be valid for norms which are implicitly subject to exceptions;
- the specificity principle can give rise to new types of circularities;
- it should be considered whether in "superior evidence" cases the solution of Loui or of this paper is adopted.
- a choice must be made between DK* or DK as the set of preferred facts, i.e. whether facts which follow from every choice in an ambiguous case should be preferred.

6. Conclusion

This paper has developed with logical tools a formal theory about preferring the most specific argument, improving other proposals in some important respects. Its main contributions to AI and Law are that (1) it offers a way to deal with exceptions to legal rules, (2) it draws part of the dividing line between hard and easy questions, which is relevant for programs which "spot issues", and (3) it provides a touchstone for evaluating the soundness and completenes of implementations of "Specific defeats general".

In conclusion, by treating arguments as internally subject to the rules of (default) logic and defining specificity in logical terms this paper has shown that logic can be useful as a tool in legal reasoning even if deduction is not regarded as the right way to model it.

Acknowledgements

This research was supported under contract no. 410-203-002 by the Foundation for research and law (NESRO) and the Foundation for computer science research (SION), both recognized by the Netherlands organization for scientific sesearch (NWO). I wish to thank John-Jules Meyer and Arend Soeteman for their valuable comments on earlier drafts of this paper.

References

Ashley and Rissland 1987: K.D. Ashley and E.L. Rissland, A case-based system for trade secrets law. Proc. 1st Int. Conf. on AI and Law, Boston 1987, 60-66.

Brewka 1989: G. Brewka, Preferred subtheories: an extended logical framework for default reasoning. Proc. IJCAI 1989, 1043-8.

Delgrande 1987: J. Delgrande, An approach to default reasoning based on a first-order conditional logic: revised report. Artificial Intelligence **36** (1988) 63-90.

Etherington and Reiter 1983: D.W. Etherington and R. Reiter, On inheritance hierarchies with exceptions. Proc. AAAI 1983, 104-108.

Gardner 1987: A. v.d. L. Gardner, An Artificial Intelligence approach to legal reasoning. MIT press 1987.

Gordon 1989: T.F. Gordon, Issue spotting in a system for searching interpretation spaces. Proc. 2nd Int. Conf. on AI and Law, Vancouver 1989, 157-164.

Loui 1987: R.P. Loui, Defeat among arguments: a system of defeasible inference. Computational Intelligence, Vol 3, no 2 (1987), 100-106.

McCarty 1988: L.T. McCarty, Programming directly in a nonmonotonic logic. Technical Report LRP-TR-21, Computer Science Department, Rutgers University, September 1988.

Nute 1989: D. Nute, Defeasible reasoning: a philosophical analysis in Prolog. In J.H. Fetzer (ed): Aspects of Artificial Intelligence. Kluwer Ac. Publ., 1988, 251-288.

Oskamp et al. 1989: A. Oskamp, R.F. Walker, J.A. Schrickx, P.H. van den Berg, Prolexs, divide and rule: a legal application. Proc. 2nd Int. Conf. on AI and Law, Vancouver 1989, 54-62.

Poole 1985: D.L. Poole, On the comparison of theories: Preferring the most specific explanation. Proc. IJCAI 1985, 144-147.

Poole 1988: D.L. Poole, A logical framework for default reasoning. Artificial Intelligence **36** (1988), 27-47.

Prakken 1991: H. Prakken, A formal theory about preferring the most specific argument. Research report Dept. of mathematics and computer science, Vrije Universiteit Amsterdam: to appear 1991.

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

Routen 1989: T. Routen, Hierarchically organized formalizations. Proc. 2nd Int. Conf. on AI and Law, Vancouver 1989, 242-250.

Touretzky 1986: D.S. Touretzky, The mathematics of inheritance systems. Pitman, London, 1986.