Legal knowledge based systems
JURIX 91
Model-based legal reasoning

The Foundation for Legal Knowledge Systems
Editors:
J.A.P.J. Breuker
R.V. De Mulder
J.C. Hage

Henry Prakken and Joost Schrickx, Isomorphic models for rules and exceptions in legislation: J.A.P.J. Breuker, R.V. De Mulder, J.C. Hage (eds.), Legal knowledge based systems JURIX 91: Model-based legal reasoning , The Foundation for Legal Knowledge Systems, Lelystad: Koninklijke Vermande, pp. 17-27, 1991 ISBN 90 6040 989 2.

More information about the JURIX foundation and its activities can be obtained by contacting the JURIX secretariat:

Mr. C.N.J. de Vey Mestdagh
University of Groningen, Faculty of Law
Oude Kijk in 't Jatstraat 26
P.O. Box 716
9700 AS Groningen
Tel: +31 50 3635790/5433
Fax: +31 50 3635603
Email: sesam@rechten.rug.nl

http://jurix.bsk.utwente.nl/

ISOMORPHIC MODELS FOR RULES AND EXCEPTIONS IN LEGISLATION

Henry Prakken and Joost Schrickx
Computer/Law Institute
Vrije Universiteit
Amsterdam
The Netherlands

Summary

This paper investigates models supporting isomorphic formalisation of separate rules and exceptions in legislation. It is shown that models which allow for nonmonotonic reasoning make possible isomorphic formalisation in cases where models based on standard logic do not. Three models of the first kind will be discussed and compared: one using applicability clauses which are assumed to hold unless the opposite can be proven; and two using conflict resolution metarules based, respectively, on notions of specificity and priority.

1. Introduction

This paper combines theoretical and practical research in automating legal reasoning, in investigating the possibility of isomorphic formalisation of exceptions in legislation. In recent years so called "isomorphic formalisation" has often been advocated as the best way to structure legislative knowledge for the purpose of building Legal Knowledge-Based Systems [Routen, 1989; Karpf, 1989; Nieuwenhuis, 1989; Bench-Capon and Coenen, 1991]: it is argued that keeping the structure of the formalisation as close as possible to the original sources benefits, among other things, validation and maintenance; for example, isomorphism is said to support a modular way of formalising, i.e. formalising a piece of information independently of the rest of the domain. However, complete isomorphism has as yet been an unobtainable goal, partly because of some peculiar structural features of legal texts. One of these features is the way exceptions are often expressed in legislation, viz. separate from the general rule. When standard reasoning techniques based on classical proposition- or predicate logic are used, this separation has to be given up in the formalisation [cf. Nieuwenhuis, 1989:62]. This, however, decreases isomorphism.

As an alternative, this paper investigates models for formalising rules and exceptions in which not isomorphism but, instead, the standard view on reasoning is given up: particularly models will be discussed which incorporate elements of so called "nonmonotonic reasoning". It will be examined to what extent these models preserve isomorphism and support modularity, but also what the consequences are of their deviation from standard reasoning techniques. This paper is about the logical aspects of *any* knowledge representation formalism, whether it uses logical formulas or not to represent knowledge. Therefore, although the examples and methods are discussed in terms of logical formulas, our investigations are also relevant for other notational schemes for representing knowledge, for example, frames or semantic nets. The only reason to use logical formulas as the syntax is that this is the best way to abstract the logical aspects (in the semantical sense) of a knowledge representation language. Finally, it should be stressed that we will argue for the possibility rather than the desirability of isomorphism.

The structure of this paper is as follows: in section 2 some examples of separate rules

and exceptions in legislation are given, which are shown to cause problems for standard reasoning techniques if isomorphism is to be preserved. In section 3, after a final unsuccessful attempt with standard techniques, three nonstandard models are presented, of which in section 4 one element, nonmonotonic reasoning, is discussed. Finally, in section 5 the merits and disadvantages of these models are discussed, both from a practical and from a theoretical point of view.

2. Obstacles for isomorphism

2.1. Terminology

For the purpose of this paper it suffices to describe isomorphism as the situation that one "source unit" is formalised in one "KB-unit" ("KB" stands for "Knowledge Base"). By a source unit we mean the smallest identifiable unit of the source from which a norm can be extracted. In general this will be a section or a subsection of a code. By a KB unit we mean a material implication or equivalence (recall that this paper is on the logical level of knowledge representation). Isomorphism is an aspect of the result of the formalisation process, and should be carefully distinguished from modularity, which is an aspect of the process of formalising itself: in the introduction it was described as formalising a source unit without having to consider other source units. In the literature several, not totally equivalent, definitions of isomorphism can be found [Bench-Capon and Coenen, 1991; Karpf, 1989; Nieuwenhuis, 1989]. However, we believe that our description captures an important element in all of them, such that their differences do not matter in the present context.
The above implies that the two following situations are violations of isomorphism: the situation that one source unit is formalised in more KB units; and the situation in which one KB unit contains concepts from more source units, unless a source unit itself refers to other source units (as in Example 2.2, 2.3 below). The main problem discussed in this paper is how to avoid the second situation when in legislation general rules and exceptions are expressed separately. We now give some examples.

2.2. Examples of separate rules and exceptions

Example 1 Section 2 of the Dutch rent act (HPW) states that this code is not applicable to rent contracts which by their nature concern a short termed usage. Since 2 HPW is not mentioned in the rest of the act, this section causes in fact all other sections of the HPW to have an implicit exception. Note that 2 HPW does not itself contradict another rule but only renders other rules under certain conditions inapplicable.

Example 2 Section 4 HPW lays down certain necessary and sufficient conditions for the possibility to change rents once a year. Section 30-(2) HPW explicitly makes an exception to this section by stating that "contrary to 4 HPW" rents included in a rent contract of before 1976 must remain unchanged for three years.

Example 3 Section 287 of the Dutch penal code (Sr) attaches a maximum penalty of fifteen years of imprisonment to killing someone on purpose, whereas section 154-(4) Sr imposes a maximum penalty of twelve years upon killing someone in a "life-and-death" duel.

Example 4 Section 1624 of the Dutch civil code (BW) declares that if a contract has features of both rent of business accommodation and another contract, and a regulation concerning the other contract type conflicts with a regulation concerning rent of

business accommodation, the latter prevails. This is a very complex example of implicit exceptions: every rule which is not about rent of business accommodation has an implicit exception for cases in which it is in conflict with a regulation concerning rent of business accommodation.

2.3. Formalisations in standard logic

specific exception clauses

If only standard reasoning techniques are used, the simplest way to formalise these examples is by using explicit specific exception clauses: this method consists of adding the negation of the condition of the exceptional rule to the antecedent of the general rule. Thus (formalisation 1 of Ex 2.1 - 2.4):

Ex 2.1:N HPW: (conditions **and** ¬shorttermedcontract)**entails** conclusion.
In this way 2 HPW is not formalised as one separate KB unit.

Ex 2.2:4 HPW: (conditions **and** ¬<1976) **entails** rentchange_possible
30-(2) HPW: (conditions **and** <1976) **entails** ¬rentchange_possible

Ex 2.3:287 Sr: (kill **and** on purpose **and** ¬life-and-death-duel) **entails** maximum = 15
154-(4) Sr: (kill **and** life-and-death-duel) **entails** maximum = 12

Ex 2.4:1624 BW: this would require for each rule possibly coming in conflict with a regulation about rent of business accommodation to have a condition ¬rent_of_business_accommodation.

Since in these formalisations several non-contiguous source units are mixed in one KB unit, it is obvious that isomorphism is lost completely. [Nieuwenhuis 1989:62] defends a similar deviation of isomorphism by arguing that thus the implicit logical structure of the statute is made explicit. However, in the next section we will assume a nonstandard logical structure of legislation, a structure which allows the separation of general rules and exceptions, and which is therefore already explicit in the statute itself. First, however, another attempt within standard logic will be discussed.

applicability clauses

At first sight there seems to be a way to retain the standard view on the logical structure of legislation by formalising in a careful way, viz. with general applicability clauses. Consider the following formalisation of Example 2.1.

Formalisation 2 of Example 2.1

2 HPW: (shorttermedcontract(c) **and** ¬(n = 2)) **entails** ¬appl(n)
N HPW: (condition($x_1 \ldots x_m$)) **and** appl(N)) **entails** conclusion($x_1 \ldots x_m$)
n is assumed to be a variable ranging only over sections of the HPW. In this paper variables are printed in lowercase and constants are denoted by capitals and numerals. Variables are assumed to be universally quantified.

Thus, since the applicability clause does not refer to any particular source unit, no source units are mixed in the formalisation. An additional advantage is that now 2

HPW is formalised as a separate KB unit.
However, this is not the whole story: with only these two rules in the KB, N HPW will never apply to a situation, since there is no way to conclude to "appl(N)". Therefore an "equivalence formula" for this predicate is needed, with as antecedent the conjunction of the negations of all ways to render a rule inapplicable:

¬shorttermedcontract(c) **entails** appl(n).

So far the formalisation is still isomorphic, but things become more complex if there is more than one way to render a rule inapplicable. Section 7-(2) HPW, for example, renders inapplicable only chapter three of the act in case of rent of a dependent apartment, which results in the following modification of the equivalence formula.

[¬shorttermedcontract(c) **or** (dependent_apartment(c) **entails** ¬In_chIII(n))] n appl(n)

Now isomorphism is lost, since in this formula concepts of several source units, viz. 2 and 7-(2) HPW, are mixed. An additional problem is that if a norm like 7-(2) HPW is found at a later time, then the old equivalence formula has to be changed, which causes the formalisation process to be unmodular.
In conclusion, this attempt to retain standard reasoning techniques gives rise to less isomorphic formalisations of legislation and decreases the modularity of the job of the knowledge engineer, particularly because of the need for an equivalence formula. Therefore, a way is needed to avoid this formula. Such a way will be discussed in the first subsection of the next section.

3. Isomorphism supporting models

3.1. Applicability clauses in nonstandard formalisms

In this subsection the idea discussed in 2.3 to use applicability clauses is further developed. It is important to realise that in principle every norm can have separate exceptions, now or in the future: therefore, every KB unit must have an applicability clause, including exceptions themselves, such as 2 HPW. Above this was ignored in order not to complicate the discussion too much, but from now on applicability clauses will be added to every rule.
As was concluded, a way has to be found to avoid the equivalence formulas. A natural way to do this is to *assume* that the applicability condition is satisfied unless there is evidence to the contrary: for example by using a nonprovability operator, such as PROLOG's Negation As Failure (which we denote by ≈). To this end an *in* applicability clause is needed, with which Example 2.1 can be formalised as follows.

Formalisation 3 of Example 2.1

2 HPW: [shorttermedcontract(c) **and** ≈inappl(2) **and** ¬(n = 2)] **entails** inappl(n)
N HPW: [condition($x_1 \ldots x_m$) and ≈inappl(N)] **entails** conclusion($x_1 \ldots x_m$)

Thus no equivalence formulas are needed, as becomes apparent when we consider a backward reasoning mechanism which tries to derive the consequent of N HPW. Assume that, apart from the inapplicability clauses, all conjuncts of the antecedents of

the rules are entered by the user. The system will then try to derive inappl(N). In doing so, it calls 2 HPW, substituting N for n, and starts a derivation for inappl(2). It again calls 2 HPW, this time substituting 2 for n; since the conjunct ¬(n = 2) cannot be satisfied and since there are no other rules for the predicate "inappl", the system cannot complete this derivation and concludes to ≈inappl(2). This makes 2 HPW "fire" for n = N, for which reason in N HPW ≈inappl(N) cannot be satisfied and its consequent cannot be derived.
Although at first sight this seems but a minor modification of the second formalisation, in fact a crucial decision has been made: the abandonment of standard logic. Standard logical inference is *monotonic*, which means that new information can never invalidate conclusions drawn from the information obtained so far. However, formalisms which offer a way to draw conclusions on the basis of the nonderivability of other conclusions do not have this property, because such conclusions can become invalid if additional information leads to the derivability of these other conclusions. Inference notions which underlie this kind of reasoning are called *nonmonotonic.* This will be discussed in more detail in the next section. For a general overview the reader is referred to [Reiter 1987]. For now it can be remarked that in the logical literature on nonmonotonic reasoning the use of general applicability clauses in nonmonotonic formalisms is a well studied technique [e.g McCarthy, 1986].

3.2. Conflict solving metarules

Assuming the applicability of a rule unless the opposite can be proven is not the only way to preserve isomorphism in case of separate exceptions. Two, related, possible alternatives are to use conflict solving metarules based, respectively, on a notion of specificity or priority, in a system which can reason with a possibly inconsistent KB. First two examples of using specificity will be discussed.

Specificity

In Example 2.4 the most isomorphic interpretation is to regard 154-(4) Sr as a rule which covers a more specific case than 287 Sr, which makes the conflict solving metaprinciple "Lex Specialis derogat legi generali" applicable [cf. Hazewinkel-Suringa, 1989:753-4]. At first sight a problem is that the element "on purpose" is not part of the formulation of 154-(4) Sr, which prevents the antecedent of 154-(4) Sr from being a specific case of the antecedent of 287 Sr. Nevertheless, according to commonsense the agreement to duel on life-and-death implies the purpose to kill. If this piece of commonsense knowledge is explicitly added to the formalisation, specificity becomes syntactically apparent: the technical details can be found in [Poole 1985] and [Prakken 1991a]. The formalisation is then as follows.

Formalisation 2 of Example 2.4

287 Sr:(kill **and** on_purpose) **entails** maximum = 15
154-(4) Sr: (kill **and** life-and-death-duel) **entails** maximum = 12
CS: life-and-death-duel **entails** on_purpose

Also Example 2.2, 2.3 may be regarded as a Lex Specialis case. In this interpretation it is formalised as follows.

Formalisation 2 of Example 2.2, 2.3

4 HPW: conditions **def** rentchange_possible
30-(2) HPW: (conditions **and** <1976) **entails** ¬rentchange_possible

Thus formalised the second rule is a "lex specialis" of the first, since the antecedent of 30-(2) HPW is a specific case of the one of 4 HPW. The incorporation of the conditions of 4 HPW in the formalisation of 30-(2) HPW is not a violation of isomorphism, since 30-(2) HPW itself refers to those conditions by means of "contrary to 4 HPW".
The idea of this way of formalising is that the system has an algorithm which checks for specificity on the basis of the syntax of the formulas involved. In the literature on nonmonotonic reasoning this has been studied by among others [Poole 1985], [Loui 1987] and, with legal applications, [Prakken 1991a].

Priorities

As a way to implement the specificity method, sometimes the use of a general priority mechanism is recommended [Brewka, 1989]. The idea is to avoid syntactical pitfalls as with 154-(4) by giving higher priority to those rules which according to the definition of specificity would prevail: thus less care with respect to the syntax of rules is needed. Furthermore, sometimes using priorities is the most natural way to deal with separate exceptions. Consider Example 2.4, which is a complex example of implicitly expressed exceptions. Its most natural interpretation is as follows: if conflicting rules apply to a case with a contract of mixed nature, 1624 BW gives rules concerning rent of business accommodation priority over rules concerning other types of contracts.
In order to reason with priorities, the system must provide for ways to express priorities between rules and to take these priorities into account in the reasoning process. In the logical literature reasoning with priorities is studied by among others [Brewka 1989] with respect to factual commonsense reasoning, and [Alchourron and Makinson 1981] and [Prakken 1991b] with respect to normative reasoning. [Routen 1989] is a limited attempt to implement this in logic metaprogramming for legal purposes.

4. Nonmonotonic reasoning

As was explained in the previous section, using a nonprovability operator introduces nonmonotonicity into the system: because conclusions depend on the failure to derive other conclusions, they become invalid as soon as new information makes it possible to derive these other conclusions. Also the specificity and the priority approaches introduce nonmonotonicity, since in these approaches the validity of a conclusion derived from a general or lower rule depends on the failure to derive the opposite conclusion from a more specific or higher rule. Should we be happy or unhappy about nonmonotonic behaviour of legal knowledge based systems?
On the one hand, monotonicity is a reassuring property of standard logic, both pragmatically and computationally. Pragmatically it is reassuring since it guarantees that inferences which are made at some point in the reasoning process stay valid even if new information is added to the system. For the user this makes, of course, monotonic conclusions the most reliable ones. Also computationally monotonic inference is desirable, since it can make use of *local* proof methods: if there is a monotonic proof for a formula **fi** from a part of the KB, this counts as a proof from the entire KB, since new information cannot invalidate this proof. Nonmonotonic proof methods, on the

other hand, are *global*, since if they have found a way to derive **fi**, they have to continue searching, because new information might invalidate nonmonotonic conclusions: in the applicability clause approach a conclusion **fi** based on the nonderivability of a formula from a part of the KB can be invalidated by a derivation of **fi** from another part of the KB; and in the metarule approaches the opposite of **fi** may be derivable from a more specific or higher rule in the KB. Obviously, global proof methods are less efficient than local ones, since for every conclusion the entire KB has to be inspected. Moreover, global proof methods can give rise to new kinds of loops [cf. Prakken, 1991a,1991b].

However, despite this intractability of nonmonotonic formalisms, monotonicity becomes a disadvantage if it is a characteristic of the domain that sometimes conclusions are drawn on the basis of the failure to derive other conclusions. And in law this seems undoubtedly the case, as has been argued by e.g [Hage 1987] and [Gordon 1988]. Furthermore, as was demonstrated in the course of our analysis, the use of nonmonotonic representation formalisms is a way to increase isomorphic formalisation of legislation, which makes it valuable if isomorphism is the aspiration. In conclusion, it can be said that the intractability of the *formalisms* is caused by the intractability of the *problems* [cf. Etherington, 1988:69].

An additional advantage of nonmonotonic formalisms is that their ability to draw conclusions under incomplete information offers an extra possibility compared to standard methods, viz. preventing irrelevant factual questions [Routen, 1989:245; Gordon, 1988:119]. If, for example, in a negation-as-failure proof for a fact Inappl(N) information about a formula **fi** is needed, then a standard reasoning mechanism has to know whether **fi** is true or false, otherwise it cannot draw any conclusion. A nonmonotonic system, on the other hand, can also draw conclusions if nothing is known about **fi**, and this ability is desirable in two cases. The first case is when it is appropriate not to ask the user for the missing information, as, for example in Example 2.1, since the concept "shorttermed contract" is worked out in a large amount of case law which is highly factual and specific in nature [see Walker et al., 1991:45]. The second case is situations in which the missing information about **fi** is asked to the users, but in which the user can give "I don't know" answers. Such an answer has the same effect as no information about **fi** without asking.

Of course, these arguments do not decide the case in favour of nonmonotonic methods; it will always be the research goals, the domain and the application which determine the most appropriate method.

5. Comparison of the models of section 3

Degree of isomorphism

The first point of comparison is the question which of the nonmonotonic models actually preserves isomorphism best. As has become apparent from the examples, legislation does not point at one single model as the most appropriate in all cases: different norms can require different most isomorphism-preserving methods of formalisation. Therefore ideally, if a legal knowledge based-system is based on isomorphic formalisation, all three models should be implemented.

Modularity

The second point of comparison is the possibility of modular formalisation. As was pointed out in section 2, isomorphism is an aspect of the result rather than of the process of formalisation, while, on the other hand, modular formalisation, which is

formalising each source unit independently of any other source units, is an aspect of the process. Modularity is often regarded as a desirable feature of the formalisation process, since it eases the job of a knowledge engineer in several ways, for instance with respect to validation [Schrickx, 1990].
In the general literature on nonmonotonic reasoning the applicability clause approach is often said to be nonmodular, since the formalised exception has to mention explicitly to which rule it is an exception, for which reason the knowledge engineer must be aware of all possible interactions between KB units [Touretzky, 1984; Poole, 1991:295]. However, in legislation often a source unit itself explicitly says which rules it renders inapplicable: consider, for instance, Example 2.1., in which 2 HPW refers to the whole HPW. And if a norm does so, it can be formalised without looking at the content of the other rules.
At first sight the specificity-approach seems to enhance modularity. This is indeed what is claimed by various proponents of this approach in the literature on nonmonotonic reasoning, e.g by [Poole 1985,1991], [Touretzky 1984] and [Loui 1987:106]. However, if Example 2.2, 2.3 is examined more closely, it appears that the decision to add the commonsense rule is the result of anticipating a possible conflict between 154-(4) and 287 Sr, since without this rule 154-(4) is syntactically not a special case of 287. This shows that the correct formalisation can only be determined after having considered the entire source, which violates modularity.
Since in the priority approach the syntax of the formulas is not as crucial as it is in the specificity approach, using priorities seems to support modularity. However, this is not always the case. Decisive is whether priorities between norms can be computed by the system on the basis of individual properties of norms, for example their place in the general legal hierarchy or their time of enactment, or whether normconflicts have in a general way been anticipated by the legislator as in 1624 BW in Example 2.4. If the first cannot be done or the second is not the case, then the priorities have to be added by the knowledge engineer, which requires a global awareness of all possible interactions between rules, which makes the formalisation process unmodular. This particularly holds if priorities are used to implement the specificity method, since then the outcome of the definition of specificity to conflicting proof pairs has to be kept in mind.
The above observations give rise to an interesting conclusion: although isomorphism of the result and modularity of the process of formalising a domain are often regarded as two sides of the same coin, we have found situations in which isomorphism is satisfied, but modularity is not.

Modularity of adding exceptions

By this specific case of modularity we mean the situation that new exceptions can be added without having to change old information. As was shown above this is not possible in the standard approaches: in the specific exception clause approach the general rule, and in the standard applicability approach the closed world axioms have to be revised each time when a new exception is added.
All three nonmonotonic methods make it possible to add new exceptions without having to *rewrite* the old rules. However, as has become apparent in the discussion of modular formalisation, the knowledge engineer has still to be *aware* of the general rules in order to formalise the exceptions or assign the priorities in the correct way, which prevents complete modularity.

Encoding of exceptionality

One of the issues in the study of nonmonotonic reasoning is whether exceptionality

should be encoded or not, i.e. whether it should be determined by inspecting the structural properties of the formulas involved, or whether these formulas must themselves express to which rules they are subject or an exception [Poole, 1991]. In the applicability approach exceptionality is obviously encoded in the formulas, whereas in the specificity approach it is not encoded at all. The priority method is somewhere in between: exceptionality is not encoded in the formulas but in the priorities. It is interesting to note that the natural language-sources sometimes encode exceptionality, as in Examples 2.1, and 2.2, 2.3, and sometimes do not, as in examples 2.4 and 2.4. For this reason, if isomorphism is to be preserved, methods with and without encoding are both needed.

Implementation

Finally, the prospects for implementation have to be considered. As was already pointed out in section 4, nonmonotonic formalisms are computationally less efficient than standard reasoning techniques. However, if the choice is made for nonmonotonicity, then the applicability clause approach with a nonprovability notion seems to have the best prospects for implementation, viz. by using negation as failure in logic programming.

6. Conclusion

In this paper we have investigated various models for isomorphic formalisation of separate rules and exceptions in legislation: assuming the applicability of rules unless the opposite can be proven, applying the "Lex Specialis" principle and using priorities. The reader might have expected a clear and definite judgement on which model is the best. This, however, is impossible: all models have advantages as well as disadvantages, and this calls for tradeoffs which can have different results with respect to different research goals, applications or domains. Nevertheless, our study has resulted in some useful conclusions.
The first is that if an alternative view on legal reasoning, leaving room for nonmonotonicity, is employed, new ways to increase isomorphism become available, because in many nonmonotonic formalisms the separation of rule and exception can be preserved. Secondly, even if the models of section 3 induce more isomorphism of the result of the formalisation process, they do not always enhance modularity of the process itself, since often the correct formalisation of a source unit depends on the content of other source units. Furthermore, methods which increase isomorphism often decrease computational efficiency; however, if the choice is made for nonmonotonicity, then the general applicability clause approach with negation as failure has the best prospects for implementation, viz. in logic programming. Finally, if isomorphism is the aim then, ideally, it is not sufficient to use one method with exclusion of the other ones, since all three methods capture a part of the formal structure of legislation.

7. Acknowledgements

The work of Henry Prakken is supported under contract no. 410-203-002 by the Foundation for research and law (NESRO) and the Foundation for computer science research (SION), both recognised by the The Netherlands organisation for scientific research (NWO). Joost Schrickx works as a knowledge engineer for the PROLEXS project, the development of a legal expert system shell, together with applications, at the Computer/Law Institute of the Vrije Universiteit Amsterdam. We would like to

thank John-Jules Meyer, Arend Soeteman and our collegues of the Computer/Law Institute for their valuable comments on earlier drafts of this paper.

8. References

[Alchourron and Makinson 1981] C.E. Alchourrón and D. Makinson, Hierarchies of regulations and their logic, in: R.Hilpinen (ed), *New studies in deontic logic*, Reidel, Dordrecht 1981, 125-148.

[Bench-Capon and Coenen 1991] Exploiting isomorphism: development of a KBS to support British Coal insurance claims. *Proceedings of the third International Conference on Artificial Intelligence and Law*, Oxford. ACM Press 1991, 62-68.

[Brewka 1989] G. Brewka, Preferred subtheories: an extended logical framework for default reasoning. *Proceedings of the eleventh Joint International Conference on Artificial Intelligence*, 1989, 1043-1048.

[Etherington 1988] D.W. Etherington, *Reasoning with incomplete information*. Pitman, London, 1988.

[Gordon 1988] T.F. Gordon, The importance of nonmonotonicity for legal reasoning. In H.Fiedler, F.Haft, R. Traunmüller (eds), *Expert systems in law*. Tübingen, 1988, 110-126.

[Hage 1987] J. C. Hage, De betekenis van niet-standaardlogica's voor juridische expertsystemen. *Computerrecht* 4 (1987), 233-9.

[Hazewinkel-Suringa 1989] D. Hazewinkel-Suringa, *Inleiding tot de studie van het Nederlandse Strafrecht*, 11th edition, revised by J. Remmelink. Samson H.D. Tjeenk Willink, Alphen aan den Rijn, 1989.

[Karpf 1989] J. Karpf, Quality assurance of legal expert systems. *Preproceedings of the third International Conference on "Logica, Informatica, Diritto"*, Florence 1989, 411-440.

[McCarthy 1986] J. McCarthy, Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence* 28(1986), 89-116.

[Nieuwenhuis 1989] M.A. Nieuwenhuis, *Tessec: een expertsysteem*

voor de Algemene Bijstandswet. Kluwer, Deventer, 1989.

[Poole 1985] D.L. Poole, On the comparison of theories: Preferring the most specific explanation. *Proceedings of the ninth International Joint Conference on Artificial Intelligence*, 1985, 144-147.

[Poole 1991] D.L. Poole, The effect of knowledge on belief: conditioning, specificity and the lottery paradox in default reasoning. *Artificial Intelligence* 49 (1991), 281-307.

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

[Prakken 1991b] H. Prakken, Reasoning with normative hierarchies. To appear in *Proceedings of the first International Workshop on Deontic Logic and Computer Science*, Amsterdam 1991.

[Reiter 1987] R. Reiter, Nonmonotonic reasoning. *Ann. Rev. Comput. Sci.* 2, 1987: 147-186.

[Routen 1989] T. Routen, Hierarchically organised formalisations. *Proceedings of the second International Conference on Artificial Intelligence and Law*, Vancouver. ACM Press 1989, 242-250.

[Schrickx 1990] J.A. Schrickx, Prolexs-formalisation, a way to enhance validation. In D. Kracht, C.N.J. de Vey Mestdagh, J.S. Svensson (eds): *Legal Knowledge based systems. An overview of criteria for validation and practical use.* Koninklijke Vermande BV, Lelystad 1990, 69-77.

[Touretzky 1984] D.S. Touretzky, Implicit ordering of defaults in inheritance systems. *Proceedings of the fifth National Conference on Artificial Intelligence*, Austin TX, 1984, 322-325.

[Walker et al. 1991] R.F. Walker, A. Oskamp, J.A. Schrickx, G.J. van Opdorp, P.H. van den Berg, Prolexs: creating law and order in a heterogeneous domain. *International Journal of Man-Machine Studies* 35 (1991), 35-67.