Legal knowledge based systems
JURIX 93
Intelligent Tools for Drafting Legislation, Computer - Supported Comparison of Law

The Foundation for Legal Knowledge Systems
Editors:
J.S. Svensson
J.G.J. Wassink
B. van Buggenhout

Nienke den Haan, Towards support tools for drafting legislation, in: J.S. Svensson, J.G.J. Wassink, B. van Buggenhout (eds.), Legal knowledge based systems JURIX 95: Intelligent Tools for Drafting Legislation, Computer - Supported Comparison of Law, Lelystad: Koninklijke Vermande, pp. 23-30, 1993 ISBN 90 5458 089 5.

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

TOWARDS SUPPORT TOOLS FOR DRAFTING LEGISLATION

NIENKE DEN HAAN

Summary

Drafting laws is not just translating do's en dont's into obligations and prohibitions. The complex structures of interrelated acts and sections have to be modelled as well. Drafting legislation exists of numerous additions and deletions of texts. When a legal drafter adds an obligation, the alteration may effect the boundaries of a prohibition. This paper discusses modelling norms, and describes methods for regulation drafting, repair and refinement. Each version is tested on a legal knowledge based system in order to compare with the desired behaviour. In this paper the design of an intelligent toolset is presented, both for drafting and refinement, as well as for application simulation.

1. Introduction

Drafting legislation resembles in a way the cyclic process of system design. Legal drafters have a common goal of intended behaviour, whereas system designers have to design a system that meets its design criteria [Breuker and Wielinga, 1989]. Software engineering cycles ensure that the system design is built up *gradually* and *correctly*. In legal drafting we want to use this method as well. The point is however, that we don't need to design an entire LKBS (legal knoweldge based system) for each new law domain. General legal problem solving behaviour has been implemented in legal knowledge based systems, for instance on the domains of industrial property [Nitta and Nagao, 1985], income tax [Sherman, 1987], benefit law [Nieuwenhuis, 1989] and traffic law [denHaan and Breuker, 1991]. The core of legal problem solving is similar for all law domains. An important difference in problem solving behaviour exists for different types of legal knowledge based systems built to perform different dedicated roles. For applying law, advice, planning and comparison specific reasoning modules are to be desired. Therefore, in the next section the design of a legal drafting environment is outlined, which is based on an existing KBS. Section 3 describes how regulations can be drafted and refined, using the representation and reasoning knowledge from the given LKBS designed for law application. Consequently, 3.2 represents the core of this research. In section 4 is shown how the relations between models and legal sources play a role in regulation refinement. The last section summarises.

2. Selecting a support environment

Drafting a certain type of legislation can only be supported when it exhibits a type of reasoning which is present in an operational LKBS. In figure 1 an architecture in this fashion for application of law is given as used in the traffic law system.
In this paper the base LKBS chosen is the TRACS architecture. In figure 2 the general design of a drafting environment is given. The legal reasoning modules of {\sc tracs} are loaded into the LKBS shell. The LKBS shell (left in figure 2) provides the drafting and testing environment for legislation under development. Only validated legal reasoning procedures are used to build the reasoning modules of an empty LKBS, because the drafters are not allowed to change the contents of standard legal reasoning modules. Any changes to overcome drafting errors have to be performed within the knowledge bases of the LKBS.

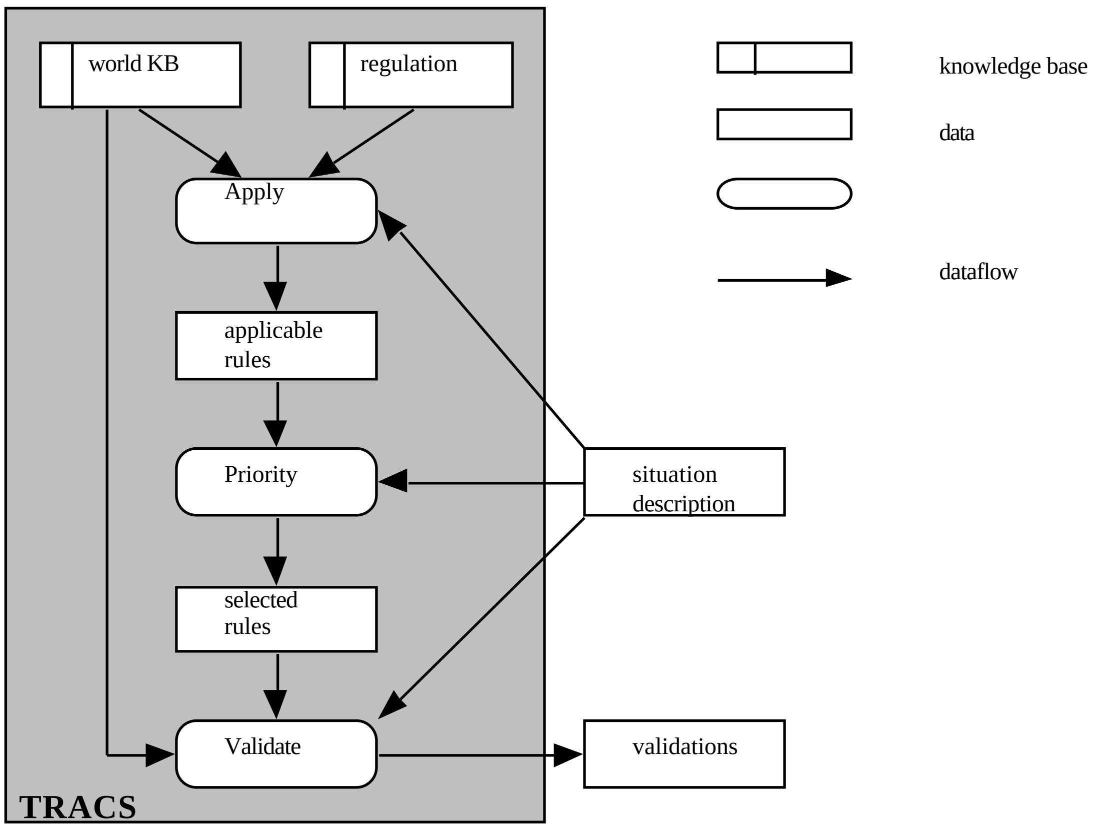

Figure 1: TRACS architecture

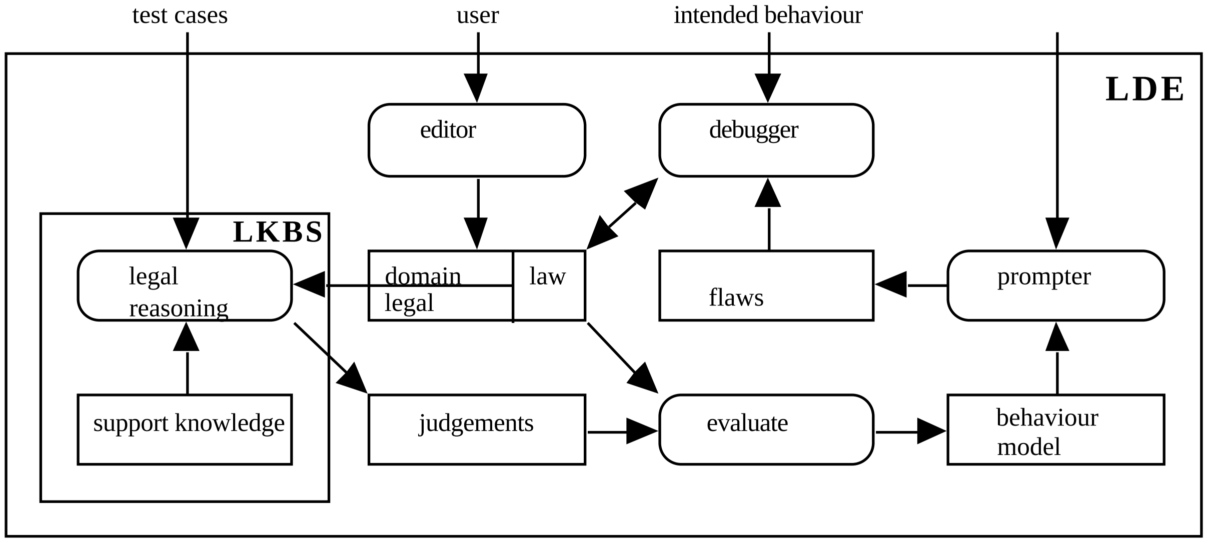

Figure 2: Design of the Legal Drafting Environment

When legislation concerning law application has to be drafted, only the legal sources have to be modelled, and moulded into the specific reasoning architecture. To accommodate the TRACS architecture to law application in a new domain, the knowledge bases have to be filled with the domain and regulation knowledge of that domain. The structure of the legal reasoning modules remains unchanged. The best way is to use an LKBS that has already proven its merits, so its reasoning modules correctly model the required type of legal reasoning. To assess its correctness, legal reasoning itself can be tested by constructing a set of test cases such that the results of law application can be predicted exactly. This type of testing scrutinises the legal reasoning structure prior to drafting and testing new legislation. For instance, an artificial regulation can be drafted, in which exceptions and exclusions are embedded. When this law is presented to legal experts in order to solve special cases, the experts have to exhibit generally the same reasoning patterns and provide exactly the same answers. In [Svensson et al., 1992] the ExpertiSZe system is described that analyses the automatic application of the social security law in order to study its consistency and social and economic impact, based on the micro-simulation method further elaborated in [Svensson, 1993]. In this approach we want to go further than testing by application. Inspired by general AI-techniques such as knowledge base refinement [Aben and van Someren, 1990], we want to give legal drafters interactive methodological support.

2.1. Using intelligent tools for drafting legislation

Separated representation of legal knowledge and world (domain) knowledge (see [Bench-Capon, 1989] and [Breuker and den Haan, 1991]) implies that the supporting intelligent modules are also divided over these two categories. There may be special dedicated tools in the fashion of Shelley [Anjewierden, et al., 1992] for editing domain information, and for editing the regulations themselves. In the former, constructs for typing information are provided, and in the latter legal expertise such as the structure of paragraphs and sections is supported. Drafting new regulations may start from scratch [den Haan, 1992]. When a law of a certain domain is to be drafted, the domain knowledge is modelled and represented in the domain knowledge base. Only world knowledge is represented here, not legal information, i.e. conceptual knowledge which is needed to *interpret* the law. If an existing law is altered, then existing representations and concepts can be used during the editing phase. When a version of the law is completed, test cases are imported into the LKBS shell. The resulting judgements are examined, and yield a model of behaviour as described by the current regulation. Only when this model is compared to the intended behaviour, i.e. the external drafting requirements, flaws can be detected. At this point, the editing/testing cycle starts again. When the modelled behaviour matches the intended behaviour, the drafting cycle has been completed, and the knowledge bases are filled with the final version.

2.2. Testing

The draft of the law text is tested by application to a set of given test cases. This yields a set of judgements. This process is exactly the same as the standard law application. The intended behaviour has been used on beforehand to determine the intended judgements the LKBS should reach. By examining the intermediate judgements and the underlying normative structures in the law, the behaviour as prescribed by the current version of the law text is created. Comparing this model to the intended behaviour may result in a series of flaws. The user can edit (new) sections, or bearing in mind the intended behaviour, the user can then debug the specific sections in the law responsible for the errors (this is described in more detail in paragraph 3). The cycle of editing, testing and debugging goes on until the judgements model equals the intended behaviour.

To test laws, a given set of situation descriptions is compared to the new regulations. The generic situations are constructed using the formulations in the regulations themselves. When all descriptions in the new regulations are gathered, this gives an overview of the real world behaviour the law can reason about. Anything outside this scope cannot be tested, because terms unknown to the system cannot be compared and related to its domain knowledge. It is also possible to generate a set of test situations. Terms in the description of the intended behaviour, possibly complemented by knowledge about the structure of legal concept types and domain concept types may give a situation generator directions about the construction of possible combinations for testing situations.

A set of applicable rules is constructed for each situation. The definition of a legal conflict is that a case gives rise to an inconsistent set of conclusions, i.e. the selected (applicable) legal rules have contradictive conclusions. Since the intended contradictions have been filtered out by the meta-rules of the LKBS, each remaining contradiction has to be studied. Repairing an unintended contradiction is guided by the meta-rules as well. Therefore, the following section first describes meta-rules for specificity as well as rule application, and then presents methods for repair and refinements.

3. Repair and refinements in regulations

Even when law texts only consist of obligations, contradictions may occur because the obliged situations themselves may be incompatible. Paragraph 3.1 suggests that some of these errors originate in faulty definitions of the meta-rules which result in the selection of the wrong section. On the other hand, paragraph 3.2 explains that also the definitions of the sections themselves may be culprits.

3.1. Controlling meta-rule application

When meta-rules solve the conflict by preferring one of the contradicting rules, an exception structure was presumably its source. In legal theory the following meta-rules are applied to solve conflicts [Hamfelt and Barklund, 1990]: Lex specialis performs a standard resolution of conflicts by selecting the most specific rule. Solutions for determining the specificity of legal rules and legal concepts can be found in the areas of set theory, taxonomic relations and abstraction hierarchies. Lex superior states that rules from distinctive law texts applying to the same area are distinguished with respect to the importance of the legislative body, and lex posterior allows that rules are selected on grounds of chronology. Lex specialis handles most of the conflicts, because exception structures are based on the notion of specificity. Therefore, operationalising lex specialis has a strong priority in modelling legal reasoning. The rules for specificity (A B: A is more specific then B) as laid out in table 1 are based largely on the notion of subsumption (A ≤ B: A is a subtype of B).

$$\frac{\alpha{:}\tau(\alpha) \qquad \beta{:}\tau(\beta) \qquad \tau(\alpha) \leq \tau(\beta)}{\alpha \leq \beta}$$

$$\frac{\begin{array}{l} M{:}\tau(M)(\alpha(1),\ldots,\alpha(\mu)) \qquad N{:}\tau(M)(\beta(1),\ldots,\beta(\nu)) \qquad \tau(M) \leq \tau(N) \\ \quad {}_{1\leq i\leq\mu,1\leq j\leq\nu}\ \alpha(i) \leq \beta(j) \end{array}}{M \leq N}$$

$$\frac{{}_{1\leq\iota\leq\mu,\ 1\leq\phi\leq\nu} A(M(\iota) \leq N(\phi) \qquad {}_{\iota,\phi}\ M(\iota) < M(\phi)}{A \quad B}$$

$$\frac{A \rightarrow B}{A \quad B} \qquad\qquad \frac{B \quad A}{A \quad B}$$

Table 1: Rules for determining specificity

A and B are the condition sets of legal rules and α, β are terms of resp. type τ(α), τ(β). Condition sets are used to determine the applicability of rules, so examining the specificity relation of two rules boils down to comparing their condition sets. Whenever conditions entail, subsume or are a subset of another condition set, then the other is more specific. The subsumption relations between condition sets are defined over predicates M,N and terms. When at least one of the constituents of a conditions set (or likewise of a predicate) subsumes a constituent in the other, on the condition that all other constituents are themselves not subsumed. The lowest level to decide subsumption is typological information which originates in the domain descriptions in the world KB. Repairs to unintended conflicts may now be solved either by changing the typological information, by changing a meta-rule or by adding a new specific meta-rule. Meta-rules may be as specific as *``In situation S select section N''.* Note that this type of mending yields a complex patchwork of repairs.

To be able to transform legal rules, their definition must be formalised:

$$x \quad M(1) \circ M(\mu) \rightarrow N \circ N(\nu)$$

where all x are bound in one of $M(1),\ldots,M(\mu),\ N,\ldots,N(\nu)$ In this formula the predicates M,N denote references to world knowledge (relations between variables or other predicates). Each predicate may have a number of arguments containing variables α. The α are typed variables, e.g. V:vehicle. The operators (o) may be either conjunctive ( ) or disjunctive ( ). The M describe the necessary conditions for a legal rule and the N contain the juridical statements about the agent(s) in question. The arrow means that the juridical statements follow necessarily when the conditions are applicable.
Table 2 gives a formalization of the determination of the applicability of rules. The rule is applicable when all its terms in the condition are supertypes of concepts in the situation (or equal). Sit(M(i)) is an abbreviation of the fact that M(i) is a member of the given situation description (case at hand), and likewise A(N(j)) for N(j) in condition set A.

$$\frac{{}_{1\leq i\leq\mu}\ \mathrm{Sit}(M(i)) \qquad {}_{1\leq j\leq\nu}\ A(N(j)) \qquad M(i) \leq N(j)}{\mathrm{applicable}(A)}$$

Table 2: Application of rules

Removing all unintended inconsistencies from law texts can be performed in a positivistic way by legal knowledge based systems that operate purely on the basis of the law text and legal meta-rules. The following paragraph shows that impasses can be solved by abstracting or specifying concepts or rules according to the lex specialis.

3.2. Specialisation and generalisation

Unintended conflicts may have occurred when in the complex structure of norms sections contain too weakly or too strongly posed elements. Their definitions are not blindly removed or altered: the definitions are reused as input to the tuning mechanisms. In knowledge acquisition research attention is currently given to the aspect of reusability. In the KADS project (ESPRIT Project P5248, KADS-II) a library of primitive inferences supports the knowledge engineer. They serve as initial building blocks in system specifications. In [Aben, 1993] a formalization of inference schemes is given, as well as tuning mechanisms to transform given inferences to meet more specific or general requirements. In this way, inferences can either be directly reused, or after (minor) alterations. The same principle holds for the definitions of sections of law, or even for larger constructs.
When a law is tested and yields unintended inconsistencies, the definitions of the meta-rules are able to guide the user to the culpable rules. To remove the arisen conflict, one of these rules has to be selected. Since the most important meta-rule handles *specificity,* one the conflicting rules has to be made more or less specific. The result of this action is that:

- The rule has been made more specific and is no longer applicable. The counteracting rule now prevails.
- The rule has been made more specific, and will be selected by lex specialis.

- The rule has been made more general. Note that a rule can never be made too general. The most general rule will be applicable to any situation. Preference will hence be given to the other rule.

To weaken or to strengthen a rule, the conditions of the rule have to be generalised or specialised, thus increasing or decreasing the grounds for applicability. For each conflict of rules, the intention of the original articles must be checked. Using the definition of the legal rule above (1) this leads to the rule modifications shown in table 3. The formulae M are the originals, and N are the modifications. M(β): α>β reads: a new β which is more loosely typed than α (α>β).

rule	formula	specialisation	generalisation
1	M(α)	M(β): α>β	M(β): α<β
2	M(α)	N(α): M>N	N(α): M<N
3	M(α) N(β)	M(α)	M(α) N(β) T(γ)
		M(β)	
		M(α) N(β)	
4	M(α) N(B)	M(α) N(β) τ(γ)	M(α)
			N(β)
			M(α) N(β)
			(M(α) N(β)) T(γ)

Table 3: Overview of modifications

In these modifications M,N and T are predicates in the condition set of a rule, and affect the application grounds of that rule. Alteration of the *conclusion* part of a rule does not lead to specialisation or generalisation, but yields less or more *normation*.
In this approach the rule-application level and the meta-level are clearly separated [vanHarmelen, 1989]. Other solutions mix these levels and try to establish preference relations for each newly found rule, e.g. [Prakken and Schricks, 1991]. Repairing an unintended conflict at the meta-level consists of defining a new local meta-rule that overcomes a specific preference problem. Due to the general impact of meta-rules, it is important to check the degree of locality of this repair immediately by applying test situations. When a regulation is tested against given situations, legal drafters can check whether the resulting law application conforms to the intended normations. The next paragraph elaborates on this aspect.

4. Intended behaviour

Laws are meant to regulate behaviour. Intended behaviour can be laid out in obligatory sections. These obligations are applicable to general situations described in the conditions of the sections. When for subclasses of situations or individuals other measurements have to be taken, the law must provide exception structures. There are two ways of establishing such exclusions. First of all, when behaviour is either obliged or prohibited, exceptions are stated by resp. prohibitions, and obligations or permissions. Secondly, when an action is obliged, an exception is formed when the opposite of that action is obliged in other situations.
In a strict logical sense, all internal contradictions in law texts yield logical inconsistencies, because they allow incompatible conclusions to be drawn from one case. However, most of the contradictions between rules are intended, because they form exception structures. All permissions are exceptions to prohibitions or obligations. A prohibition may be an exception to an obligation when it is more specific and vice versa. Active search for inconsistencies in law texts will uncover many instances. However, not all inconsistencies can be found directly in the law text, additional knowledge about the legal world is also necessary. Different interpretations of concepts can also produce conflicts. Judges have the right to decide between any contradictive rules, and can thus solve conflicts based on inconsistencies or vague terms. The meaning and goal of a law

text must always guide this process, because in this way sometimes repairs of unintended conflicts are proposed.

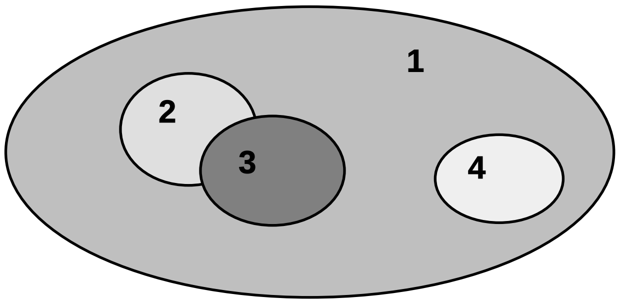

Figure 3: Intended behaviour

In the example in figure 3, the initial draft stated that the students of the Department of Computer Science and Law were allowed to use the computers. It soon turned out that this rule was too general, and we didn't want our students to be hacking around violating privacy. This behaviour was forbidden (2). Playing games is not part of the intended behaviour and was also excluded (3). The figure shows that there is some overlap between these two categories, e.g. writing supposedly funny messages to other users' screens. Another option would have been to widen the definition of the word hacking to game playing. Furthermore, it was decided that commercial use was to be forbidden on university equipment (4). Since commercial use does not fit under hacking or games, a new section had to be drafted. At this point more and more exceptions to the general rule (1) seem to be emerging. An optional solution would be to narrow the application area of the general rule by using an extra descriptor, e.g. to `students are allowed to use the computer for educational purposes'.

5. Summary

At all intermediate stages of legal drafting the new artefact must be tested. This article proposes to perform testing by means of a legal expert system shell. The application is then compared to the desired behaviour. When a law text contains conflicting rules that yield unintended inconsistencies, the definitions of these rules will have to be altered. Section 3 described how sections can be tuned in order to eliminate the conflict. Apart from the intended behaviour to be modelled, also side-effects of all the intermediate updates have to be controlled. The tuning mechanism provides a testing environment for optimalization of existing laws, as well as for drafting new laws. If law texts would yield a minimal number of unintended conflicts, then correct procedures would give rise to less retrials and appeals, thus decreasing the cost of law enforcement. Regulation refinement is especially important when individuals have found loopholes in the law text. Pure positivistic legal reasoning is only successful when law texts contain no combinations of legal rules that yield unintended conflicts. The situation where legal reasoning cannot offer a solution is highly undesirable, because laws are required to render unambiguous judgements.

The advantages of a legal knowledge based system which is highly modular and clearly separates all types of knowledge involved are also noticeable in the area of drafting law. In this paper an environment is described in which the shell of such an LBKS is used as an experimenting tool for (new drafts of) a regulation. Further support is provided by dedicated legal editors and browsers. A version control mechanism will be added to the toolset. When an alteration appears to do more harm than good in one of the following test cycles, the records of changes on the rules then enable the user to retract some of his/her tuning steps.

Legal knowledge based systems can only support drafting legislation on the condition that the correctness of their legal reasoning modules has already been determined. Most legal expert systems are designed to operationalize the application of law. The advantage these LKBSs offer are uniform legal reasoning and judgements. When they are used for experiments on law texts, they will therefore always give predictable and stable results. An LKBS which performs its reasoning tasks in the same fashion as legists and contains a description of the law which is isomorphic to the original regulation text provides the best insight to the legal drafters using the automated drafting environment.

References

[Aben and van Someren, 1990] M. Aben and M.W. van Someren. Heuristic refinement of logic programs. In L.C. Aiello, editor, *Proceedings ECAI--90*, pages 7-12. Pitman, 1990.

[Aben, 1993] M. Aben. Formally Specifying Re-usable Knowledge Model Components. *Knowledge Acquisition Journal*}, 1993. Forthcoming.

[Anjewierden et al., 1992] A. Anjewierden, J. Wielemaker, and C. Toussaint. Shelley - computer aided knowledge engineering. *Knowledge Acquisition*, 4(1), 1992. Special issue “The KADS approach to knowledge engineering”.

[Bench-Capon, 1989] T.J.M. Bench-Capon. Deep models, normative reasoning and legal expert systems. In *Proceedings of the 2nd International Conference on AI and Law*, Vancouver, 1989. ACM.

[Breuker and denHaan, 1991] J.A. Breuker and N. den Haan. Separating world and regulation knowledge: where is the logic? In *Proceedings of the third international conference on AI and Law*, Oxford, 1991. ACM Publishers.

[Breuker and Wielinga, 1989] J.A. Breuker and B. Wielinga. Models of expertise in knowledge acquisition. In *Topics in Expert System Design: Methodologies and Tools*. North Holland, 1989.

[den Haan and Breuker, 1991] N. den Haan and J.A. Breuker. A Tractable Juridical KBS for Teaching and Applying Traffic Rules. In *Model-Based Legal Reasoning - Fourth International Conference on Legal Knowledge Based Systems, JURIX-91*, pages 5--16, Lelystad, December 1991. Koninklijke Vermande.

[den Haan, 1992] N. den Haan. TRACS: A Support Tool for Drafting and Testing Law. In *Information Technology and Law -- Fifth International Conference on Legal Knowledge Based Systems, JURIX-92*. Koninklijke Vermande, December 1992.

[Hamfelt and Barklund, 1990] A. Hamfelt and J. Barklund. Meta- programming for representation of legal principles. In *Proceedings Meta90*, pages 105-122. Katholieke Universiteit Leuven, 1990.

[Nieuwenhuis, 1989] M.A. Nieuwenhuis. TESSEC, een expertsysteem voor de Algemene Bijstandswet. Kluwer, Deventer, NL, 1989.

[Nitta and Nagao, 1985] K. Nitta and J. Nagao. Krip: a knowledge representation system for laws relating to industrial property. In *Logic Programming*, pages 276--286. Electrotechnical Laboratory and Nihon Business Consultant, Springer Verlag, 1985.

[Prakken and Schricks, 1991] H. Prakken and J. Schricks. Isomorphic Models for Rules and Exceptions. In J.A. Breuker, R.V. de Mulder, and J.C. Hage, editors, *Proceedings of the fourth International Conference on Legal Knowledge Based Systems: Model-Based Legal Reasoning*, pages 17-27, Lelystad, December 1991. Koninklijke Vermande.

[Sherman, 1987] D.M. Sherman. A prolog model of the income tax act of canada. In *Proceedings of the First International Conference on Artificial Intelligence and Law*, pages 127- 136, Boston, May 1987.

[Svensson, 1993] J.S. Svensson. Kennisgebaseerde microsimulatie. PhD thesis, Universiteit Twente, 1993.

[Svensson et al., 1992] J.S. Svensson, P.J.M. Kordelaar, J.G.J. Wassink, and G.J. van 't Eind. ExpertiSZe, a Tool for Determining the Effects of Social Security Information. In C.A.F.M {Grütters}, J.A.P.J. Breuker, H.J. van den Heerik, A.H.J. Schmidt, and C.N.J. de Vey Mestdagh, editors, *Proceedings of Legal Knowledge Based Systems: Information Technology and Law, JURIX'92*, pages 51--61, Lelystad, NL, December 1992. Koninklijke Vermande.

[vanHarmelen, 1989] F. van Harmelen. *Meta-level Inference Systems*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 1989. Published in Research Notes in AI Series, Pitmann, 1991.