13th BILETA Conference: 'The Changing Jurisdiction'

Friday, March 27th & Saturday, March 28th, 1998.
Trinity College, Dublin.

Implicit Assumptions in Legal Knowledge Systems

Pepijn R.S. Visser
LIAL - Legal Informatics at Liverpool
Department of Computer Science
University of Liverpool -
PO Box 147, Liverpool, L69 7ZF
pepijn@csc.liv.ac.uk

1. Introduction

n this article we address the issue of implicit assumptions in legal knowledge systems (LKBS). We illustrate that LKBSs have many underlying assumptions mostly arising during the design process. We divide these implicit assumptions over three categories: (1) assumptions about the user, (2) assumptions about the task, and (3) assumptions about the domain. Making assumptions in the design of a LKBS cannot be avoided but both the systems designer and the users of the LKBS should at least be aware of having made the assumptions. We deem a principled method for the development of LKBSs to be a prerequisite for adequate assumption handling. In this method, ontologies fulfil the role of explicit documents of the otherwise implicit assumptions.

The remainder of this article is structured as follows. Section 2 illustrates the different kinds of assumptions that are made during the design of a LKBS. Section 3 then briefly discusses a principled method for the design of LKBSs. Section 4 elaborates on ontologies, in which certain assumptions about the designed system can be documented. In section 5 we draw conclusions.

2. Implicit Assumptions in LKBS

Before any piece of system code is written knowledge engineers and system designer have made many assumptions about a variety of aspects of the future system. Some of these assumptions are explicitly documented but a vast number of assumptions usually remain implicit. Even for rather trivial matters, such as the choice of predicate names, several implicit assumptions are made. Suppose a system designer has used predicate right(X,Y). Without explanation this term may have many different interpretations. For instance, if we look up the term `right' in WordNet we find a total of 36 different meanings. Eight of these are interpretations of the term as a noun, four are interpretations of the term as a verb, fourteen are interpretations of the term as an adjective and the remaining ten are interpretations of the term as an adverb. It could be argued that, when used in a LKBS, some of these 36 different meanings can be ruled out given the context in which the term occurs (e.g., right as opposed to left). Also, it can be argued that some meanings can, for the purpose of a particular LKBS, be considered equal (e.g., right in conformance with justice or right in conformance with the law *(1)*). Although this might reduce the number of likely interpretations there may be still many possible interpretations left. It is, for instance, not clear from the context whether a distinction should be made between `right' according to social standards and `right' according to

legal standards.

It should be noted that the WordNet ontology is not tailored to the legal domain. If we consider the word right in a legal context, and, in relation to other terms, then the number of possible interpretations is even larger. Allen (1997), for instance, illustrates that with relative a simple legal language an infinite number of legal relations can be identified. The conclusion is that there is a risk in simply assuming that terms are interpreted in their correct sense. We cannot assume that the interpretation of the knowledge engineer is the same as the interpretation of the user who has to use the system.

Below, we look at some other examples of implicit assumptions. We divide the assumptions over three categories, namely implicit assumptions about the intended user (section 2.1), assumptions about the tasks (section 2.2), and assumptions about the domain (section 2.3).

2.1 Implicit Assumptions about the User

The first category of implicit assumptions concerns the user. Bench-Capon (1991, p.190) has illustrated that many LKBSs have been designed with a specific type of users in mind. In particular, he claimed that they are designed to be used by professionals who are able to bring both an understanding of the domain and some knowledge of computers to their interaction with the system. The professionals know what the system is designed to do as well as their limitations. They know, for instance, the correct interpretation of the input terms (as in the introduction of section 2), and, the kind of questions the system is able answer (rather than then the type of questions they want answers for). Bench-Capon argues that the implicit assumptions about the intended user have to be made explicit for successful applications.

2.2 Implicit Assumptions about the Task

A second category of implicit assumptions concerns assumptions about the task of the system. Knowledge specifications are tailored to the task they are made for. This constitutes a problem since it may impede the knowledge specification to be used for other task. The problem is known as the interaction problem (Bylander and Chandrasekaran, 1987), referring to the interaction between task (purpose) and the knowledge specification. The core of the problem is that the knowledge specification has task-related assumptions that are largely implicit. Visser (1995) lists four main reasons why knowledge specifications have implicit assumptions, all arising from the development process of a specification. Below, we discuss each of these categories in turn (for a more elaborate discussion we refer to Visser, 1995).

Selections: The first reason why knowledge specifications contain implicit task-related assumptions is that during the specification process selections are made. The domain that is modelled has features that are relevant and features that are not relevant for the task at hand. Features assumed relevant are specified, features assumed not relevant are not specified. The process of making such selections is largely implicit. Apart from relevancy there are pragmatic reasons to leave out some kinds of knowledge from a LKBS. Limited availability of resources, such as time, money, labour, may all cause fragments of knowledge to be left out to the system. Which knowledge can reasonably be left out, however, is determined by the envisaged task of the future system.

Simplifications: The second reason why knowledge specifications contain implicit task-related assumptions is that during the specification process simplifications are made. As with selections, making simplifications involves leaving out certain features of the knowledge being modelled. The simplification of a feature of an entity implies that the features are modelled, but as if they were less complex or less detailed than they are in reality. We assume that simplifying a feature can only be done after it has been selected for modelling. Simplifications are considered task-related assumptions for the same reason as selections: the simplifications allowed for a particular task may be inadequate

for other kinds of tasks. Two major categories of simplifications are granularity restrictions (viz. describing the knowledge at a generalised level), and restrictions in the dynamics of the system (e.g., assuming a problem case can be described as if it happens at a single point in time).

Structuring: The third reason why knowledge specifications contain implicit task-related assumptions is that during specification process, knowledge is structured in a task-specific way. One piece of knowledge can be structured in many different ways, but many knowledge specifications can be illustrated to be structured according to the problem-solving tasks that are to be supported. The structure of a knowledge specification hides assumptions and may prevent it from being reused for other problem-solving tasks. Knowledge specifications are often created by instantiating a set of predefined structures, such as frames and rules, that are thought to resemble the conceptual patterns of human problem solvers (Newell and Simon, 1972; Minsky, 1974). The instantiation of predefined structures involves the distribution of the domain knowledge over these structures. This can be done in multiple ways, depending on the role of the domain knowledge in the system (McDermott, 1988; Schreiber, 1992). Production rules, for instance, can be show to have a `direction of reasoning', as was shown convincingly in the MYCIN experiments. Frame formalisms, to a lesser extent have the same characteristic, the envisaged use of the system determines how the domain knowledge is modelled in the frame structures.

Compilations: The last and perhaps most important reason why knowledge specifications contain implicit task-related assumptions is that during the specification process different kinds of knowledge are compiled into the representation primitives (such as frames, production rules, predicates). Lenat and Guha (1990, p.17) refer to compilations of knowledge in what they call the 'representation trap': "choosing a set of long, complex primitives (predicate names) that have a lot of knowledge compiled within them, and writing rules that are also tailored to the program's domain (omitting premises that needn't be worried about in that particular task)".

In the development of a specification different kinds of knowledge are used. The knowledge differs not only with respect to its nature (e.g., empirical, normative, heuristic, commonsense) but also with respect to the sources it is derived from (e.g., experts, text books, observations). Often, during the specification process these different kinds of knowledge are not specified separately but compiled into the representation primitives (Bench-Capon, 1987; Bench-Capon, 1989; Steels, 1990). These primitives therefore contain compilations of knowledge. Bench-Capon and Coenen (1982) discuss a knowledge analysis method from which we derive the following examples:

BuildingStructureInSafeCondition and
CompliesWithWorkPlaceRegs

Knowledge compilations are contained in specifications for two reasons. The first reason is its inevitability. During the development of a knowledge system one of the main problems is the knowledge acquisition: expert knowledge is characterised by inaccessibility and incompleteness. The problem is known as the knowledge acquisition bottleneck Kidd, 1987; Breuker and Wielinga, 1987). Knowledge from domain experts often only is available in compiled form. In the legal domain, for instance, Bench-Capon remarks with respect to expert knowledge: "It is the marshalling of this diverse material into a form where he [PRSV: the legal expert] can apply it to a case that corresponds to the development of empirical associations in the diagnostic domains" (Bench-Capon, 1989, p.39). If knowledge can only be obtained from an expert in compiled form, then knowledge compilations in specifications are inevitable. The second reason why specifications contain knowledge compilations is that it can be very efficient to use knowledge compilations (e.g., Dekneuvel, 1994). This holds in particular for shortcuts (viz. knowledge compilations that cut short frequently applied reasoning steps). The attractiveness of shortcuts in the legal domain has been stressed, for instance, by Johnson and Mead: "(...) shortcuts in the modelling of legislation are inviting and usually reliable in small systems. (...) the incorporation of each step in the legislative path appears pointless and extremely tedious to the rulebase developer" (Johnson and Mead, 1991).

Regardless of their inevitability and efficiency, knowledge compilations contain implicit assumptions. The assumptions are tacit and hence the knowledge compiled into the specification is no longer separately addressable (e.g., Clancey, 1985, p.292; Steels, 1990, p.34). This means that the knowledge compilation cannot be used for problem-solving tasks that need to have access to the fragments of knowledge underlying the compilation.

2.3 Implicit Assumptions about the Domain

The third category of implicit assumptions underlying knowledge specifications stems from the way the domain is conceptualised. Building a LKBS requires a conceptualisation of the domain, be it implicit or explicit. Such a conceptualisation consists of a set of concepts, their attributes and relations between these concepts and determines to a large extent what kind of knowledge can be represented. It has been illustrated that people conceptualise the legal domain in different ways, even if they make these conceptualisations for the same purpose (Visser and Winkels, 1997; Visser and Bench-Capon, 1998).

Sometimes, the discussion on whether certain aspects of the legal domain should be conceptualised is done in an explicit manner. Bench-Capon (1989), for instance, argues that for many LKBS one does not need to distinguish a legal modality, such as `ought' to or `permitted' (thus eliminating the need for deontic logics). The legal modality of a LKBS can be derived from the context the system is operating in.

The vast majority of assumptions on the domain conceptualisation, however, are hardly ever explicitly discussed. The assumptions are made during the process of creating the specification without much consideration. As an example we discuss two possible formalisations of a fragment of UK Social Security Law (see also: Bench-Capon and Visser, 1997). These laws state that `pensionable age' is attained at 65 in the case of a man and at 60 in the case of a woman. In a LKBS, which uses logic programming as its method, this might be represented as:

A1 pensionable_age(X) :- sex(X, male), age(X, A), A >= 65.
A2 pensionable_age(X) :- age(X, A), A >= 60.

Alternatively, this might represented as:

B1 pensionable_age(X) :- sex(X, male), age(X, A), A > 64.
B2 pensionable_age(X) :- sex(X, female), age(X, A), A > 59.

If we make the right assumptions these two representations of the concept are equivalent. But, of course, if these assumptions are violated, differences emerge.

1. A1 and A2 supposes that anyone who fails sex(X, male) will be female. This embodies two assumptions: that the second argument of sex can only be either male or female, and that sex (X, male) will fail only if X is female, not, for example, because, the sex of X is unknown.
2. B1 and B2 rely on the assumption that A will be bound to an integer. If ages were recorded as reals to express part years of completed life, the comparisons in B1 and B2 would be incorrect.
3. A1 and A2 rely on a particular execution strategy, which means that A2 will be reached only after A1 has failed. Thus, A2 contains the implicit condition that sex(X, male) fails.

There is another point here, however. The form of representation used implies a commitment to a certain conceptualisation at a high level. In the particular example, a concept, pensionable age is defined, by giving a pair of sufficient conditions, which are intended to be taken together as supplying a necessary condition. The system is intended to be executed by deducing the applicability of the defined predicates from the applicability of certain more primitive predicates. We need

therefore to be aware of the different levels of assumptions.

3. Ontologies as Explicit Assumptions

The implicit assumptions underlying a knowledge specification make the LKBS hard to maintain and reuse. The system is tailored to a specific implicit context but it is not clear what exactly constitutes this context. Consequently, it is not clear whether a system can be used in a slightly different context. In an effort to make the implicit assumptions explicit, and therewith enhance the maintainability and reusability, we adopt ontologies. An ontology is defined as an explicit specification of a conceptualisation (Gruber, 1995) and can be used to document the implicit user, task, and domain assumptions. Below, we briefly review some existing work concerning the most important of these, namely, legal task and domain ontologies.

Legal Task Ontologies

A legal task ontology describes a task typical for the legal domain from a generic perspective. It thus identifies the information in the domain that is required to perform the task. The legal domain has seen few attempts to describe tasks that are tailored to this domain. Without attempting to be exhaustive we mention five examples. Visser and Van Kralingen (1991) have described a generic task model tailored to reasoning about definitions in statutes. Valente (1995) and Visser (1995) both have addressed the description of generic legal planning systems. Quast et al. (1996) have addressed generic task models for the interpretation of vague norms. Finally, Matthijssen (1996) defined a database (hyper)indexing mechanism that is based on the objection procedure in (Dutch) administrative law.

Legal Domain Ontologies

As with task ontologies, the legal domain has seen few attempts to describe domain ontologies. Recently, research in the field of AI and Law has shown a growing interest in this field. In July 1997 the first workshop on this topic - LEGONT `97 - was held (Visser and Winkels, 1997). We here mention four legal ontologies that have gained some attention in the literature: (a) McCarty's LLD, (b) Stamper's NORMA, (c) Valente's functional ontology of law, and (d) Van Kralingen & Visser's frame-based ontology.

(a) McCarty's LLD

McCarty (1989) has proposed a language for legal discourse (lld). The basic components of lld are atomic formulae and rules. Together they allow the creation first-order expressions. Modalities, such as time and permissions, are stated as second-order expressions.

Atomic formulae are merely predicate relations used to express factual assertions, such as 'O1 is the ownership of actor A having property P', and, 'company C has issued stocks S'. A distinction is made between count terms (to express tangible objects, such as houses, and persons) and mass terms (to express intangible objects, such as cash, and stock). Rules are formed by connecting atomic formulae with logical connectives. They have a left-hand side which is an atomic formula, and a right-hand side which is a compound expression. Modalities are stated as second-order expressions. Currently, the following modalities are supported: time, events and actions, and deontic expressions (McCarty, 1989). To express temporal statements lld recognises states.

(b) Stamper's Norma Formalism

Stamper has criticised the use of traditional logics for the representation of (legal) knowledge because they suffer from some important semantic problems (Stamper, 1991). Briefly stated, traditional logics rely on symbolic representations that have only a very weak connection to the real-

world concepts they intend to denote. Stamper argues that there is need to escape from the frame of reference within which the classical logic is created (Stamper, 1991, p.229). The main ontological concepts are (a) agents, (b) behavioural invariants, and (c) realisations.

An agent is an organism standing at the centre of reality. It gains knowledge, regulates, and modifies the world by means of actions. A behavioural invariant is a description (e.g., using verbs, nouns, or adjectives) of a `situation' whose features remain invariant. Here, a situation loosely denotes some knowledge of the world, such as an object (e.g., a cup, a piano) or a state of affairs (e.g., walking, paying). Agents realise situations by performing actions. The realisation of a situation - a realisation - is specified as the combination of (1) an agent and (2) a behavioural invariant, shortly written as Ax (the situation, denoted by behavioural invariant x, that is realised by agent A). An example of a realisation Ax is John walks.

(c) Valente's Functional Ontology of Law

Valente's ontology of law (1995) is based on a functional perspective of the legal system. The legal system is considered an instrument to change or influence society in specific directions, determined by social goals. Its main function is reacting to social behaviour. This main function can be decomposed into six primitive functions, each corresponding with a category of primitive legal knowledge in lfu. Accordingly, lfu distinguishes six categories of legal knowledge: (a) normative knowledge, (b) world knowledge, (c) responsibility knowledge, (d) reactive knowledge, (e) meta-legal knowledge, and (f) creative knowledge.

Normative knowledge is characterised as knowledge that defines a standard of social behaviour. It prescribes behaviour of the people in society. The standard is defined by issuing individual norms, expressing what ought to be the case. World knowledge is legal knowledge that describes the world that is being regulated. It delineates the possible behaviour of (people, and institutions) in society, and thereby it provides a framework to define what behaviour ought (and ought not) to be performed. Responsibility knowledge is legal knowledge that either extends or restricts the responsibility of an agent for its behaviour. Reactive knowledge is legal knowledge that specifies which reaction should be taken (and how) if an agent violates a primary norm. Usually, this reaction is a sanction but it can be a reward as well. Meta-legal knowledge is legal knowledge about legal knowledge, or, legal knowledge that refers to other legal knowledge. It regulates the dynamics of the legal system, for instance, by prescribing how to make amendments and how to issue new primary norms) and it provides mechanisms to solve conflicts between instances of legal knowledge. Finally, creative knowledge is legal knowledge that allows the creation of previously non-existent legal entities. It usually is stated in imperative terms, designating an entity (e.g. a governmental committee, or, a contract between two parties) that previously did not exist to come into being from a certain point of time.

(d) Van Kralingen's and Visser's frame-based ontology

Van Kralingen (1995) and Visser (1995) have studied development techniques for legal knowledge systems. One of the ideas underlying their work is that robust (conceptual and formal) ontologies of the legal domain are necessities for reducing the task-dependency of legal knowledge specifications.

The main ontological distinction in fbo concerns the legal ontology and the statute-specific ontology. The distinction is based on the observation that some parts of an ontology are reusable across different legal subdomains. The legal ontology divides legal knowledge over three distinct entities: norms, acts and concept descriptions. For each of these entities the ontology defines a frame structure that lists all attributes relevant for the entity. Norms are the general rules, standards and principles of behaviour that subjects of law are enjoined to comply with. Norms are defined by a norm identifier, a norm type, a promulgation, a scope, conditions of application, a norm subject, a legal modality, and an act identifier. Acts represent the dynamic aspects which effect changes in the

state of the world. There are events and processes. Events represent an instantaneous change between two states, while processes have duration. The second distinction is between institutional acts and physical acts. Acts are assumed to have an act identifier, a promulgation, a scope, an agent, a type, a modality of means, a modality of manner, temporal aspects, spatial aspects, circumstantial aspects, a cause, an aim, an intentionality, and a final state. Finally, concept descriptions deal with the meanings of the concepts found in the domain. Concept descriptions have a concept, a type, a priority, a promulgation, a scope, conditions, and instances. The statute-specific ontology consists of predicate relations that are used to complement the terminology for norms, acts and concept descriptions. It defines the vocabulary with which the knowledge base is constructed.

4. Ontologies in the Development of LKBSs

The ontologies described in the previous section can be used during the system design. In this section we briefly describe how the ontologies can be used in the design process. Our starting point is a system-development method that is tailored to the specification of LKBSs (Visser et al., 1997). The method is in essence the CommonKADS methodology for knowledge system design (Breuker and Van de Velde, 1994) extended with KANT to obtain domain specifications (Bench-Capon, 1991; Bench-Capon and Coenen, 1992). The method distinguishes four design phases: (1) analysis phase, (2) conceptual modelling phase, (3) formal modelling phase, and (4) implementation phase.

As in CommonKADS there is an informal and a formal expertise model - specified in phases 2 and 3, respectively. Domain knowledge (specifying the static knowledge in the domain) is separated from control knowledge (specifying how the domain knowledge is applied to achieve a goal). Control knowledge consists of specifications of inferences (primitive reasoning steps), and tasks (a control structure over tasks and inferences) (2).

1. Analysis phase

Task identification: Identify the task(s) that have to be performed. This involves identifying potential (knowledge of) users and dividing the tasks between user and LKBS. The result of this step is description of the input, the output, and the problem-solving goals of the LKBS.

Domain identification: Identify the legal knowledge that is to be contained in the LKBS in terms of references to legal sources. Together, 1a and 1b, are meant to determine the competence of the LKBS.

2. Conceptual modelling phase

a. Method description *(4)* : Provide an informal description of how the system will perform the task by selecting an ontology of the task, or, indeed by creating one.

b. Domain ontology selection and adaptation: Select an appropriate legal ontology and tailor the ontology - if necessary - to support the tasks and methods described in the previous step. If there is no suitable domain ontology available it has to be developed.

c. Knowledge acquisition and modelling: Model the domain knowledge in accordance with the two ontologies. The result of this step is the conceptual domain specification.

3. Formal modelling phase

a. Determine boundaries of control and domain knowledge: Identify procedural knowledge embedded in the conceptual domain specification and decide whether to model this knowledge

in the expertise model as domain knowledge or as control knowledge.

b. Define control knowledge: Create a formal description of the tasks, identified in the steps 1b and 2a. This description should specify the hierarchical decomposition of tasks, and the information that is passed between tasks in a formal language.

c. Create statute-specific ontology: This step is aimed at determining and defining the predicate relations that are used to express the domain knowledge in a formal language.

d. Formalise domain knowledge: Model the knowledge described in the informal conceptual domain model by bringing together the formal ontology and the statute-specific ontology. This step results in the formal domain specification.

e. Define inferences: Define the inferences (primitive tasks) that link the control knowledge and the domain specification.

4. Implementation phase

a. Select language and platform: Select an appropriate language and platform to implement the formal descriptions of the tasks and inferences, and the domain specification.

b. Implementation: Implement the formal model in the chosen language on the chosen platform.

5. Conclusion

Any LKBS is the result of a design process during which a variety of assumptions is made about the system. We have discussed some prototypical examples divided into three categories: user, task and domain assumptions. In the course of a design process the documentation of all design assumptions is usually not feasible, nor necessary for that matter. However, there is a potential hazard in not making the assumptions explicit. The set of assumptions makes up a context in which the system can successfully operate. If the system is taken out of this context, for instance, by providing it to different types of users, using it for different tasks, or applying it to slightly different legal domains, its adequacy has to be questioned. Otherwise stated, the system has a limited reusability.

To enhance the reusability of systems, or fragments of systems, system design has to be done using recognised design phases. System design methods such as the one discussed in section 3, can be used to identify in an early stage where and when assumptions are made. Moreover, they provide a systematic account of the assumptions that are made. In system reuse and maintenance such documentation proves beneficial. The use of ontologies in the design methods, such as domain and task ontologies, facilitates sharing sets of assumptions between different people, within and between different design projects.

References

- Allen, L.E. (1997). The Language of LEGAL RELATIONS (LLR): Useful in a Legal Ontologist's Toolkit?, First International Workshop on Legal Ontologies (LEGONT'97), P.R.S. Visser and R.G.F. Winkels (eds.), Melbourne, Australia, pp.47-60.
- Bench-Capon, T.J.M. (1987). Support for Policy Makers: Formulating Legislation with the Aid of Logical Models, Proceedings of the First International Conference on Artificial Intelligence and Law (ICAIL '87), pp.181-189, Boston, Massachusetts, United States.
- Bench-Capon, T.J.M. (1989). Deep Models, Normative Reasoning and Legal Expert Systems,

Proceedings of the Second International Conference on Artificial Intelligence and Law (ICAIL `98), pp.37-45, Vancouver, Canada.
- Bench-Capon, T.J.M. (1991). Knowledge-Based Systems and Legal Applications, Academic Press, London, APIC series, No 36.
- Bench-Capon, T.J.M., and F.P. Coenen (1992). Isomorphism and Legal Knowledge Based Systems, Artificial Intelligence and Law, Vol. 1, No. 1, pp.65-86.
- Bench-Capon, T.J.M., and P.R.S. Visser (1997). Ontologies in Legal Information Systems; The Need for Explicit Specifications of Domain Conceptualisations, Proceedings of the Sixth International Conference on Artificial Intelligence and Law (ICAIL'97), Melbourne, Australia.
- Breuker, J.A., and B.J. Wielinga (1987). Use of Models in the Interpretation of Verbal Data, Knowledge Acquisition for Expert Systems, A Practical Handbook, Plenum Press, New York, United States.
- Breuker, J.A., and W. van de Velde (1994). CommonKADS Library for Expertise Modelling, Reusable Problem Solving Components, IOS Press, Amsterdam, the Netherlands.
- Bylander, T., and B. Chandrasekaran (1987). Generic tasks for knowledge-based reasoning: the "right" level of abstraction for knowledge acquisition, International Journal of Man-Machine Studies, Vol. 26, pp.231- 243.
- Clancey, W.J. (1985). Heuristic Classification, Artificial Intelligence, Vol. 27, p.298-350.
- Dekneuvel, E., M. Ghallab, and H. Philippe (1994). Distributed Inference on Compiled Knowledge for Real Time Distributed Systems, Proceedings 11th European Conference on Artificial Intelligence, A. Cohn (ed.), pp.719-723, Joh Wiley & Sons, Ltd., New York, United States.
- Gruber, T.R. (1995). Toward principles for the Design of Ontologies Used for Knowledge Sharing, Int. Journal of Human-Computer Studies, Vol.43, pp.907-928.
- Johnson, P., and D. Mead (1991). Legislative Knowledge Based Systems for Public Administration - Some Practical Issues, Proceedings of the Third International Conference on Artificial Intelligence and Law (ICAIL `91), pp.108-117, Oxford, England.
- Kidd, A.L. (1987). Knowledge Acquisition for Expert Systems, A Practical Handbook, Plenum Press, New York, United States.
- Kralingen, R.W. van (1995). Frame-based Conceptual Models of Statute Law, Computer/Law Series, No. 16, Kluwer Law International, The Hague, The Netherlands.
- Lenat, D.B., and R.V. Guha (1990). Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project, Addison-Wesley, Reading, Massachusetts, United States.
- Matthijssen, L. (1996). A Task-Based Hyperindex for Legal Databases, Legal Knowledge Based Systems; Foundations of Legal Knowledge Systems (JURIX `96), pp.59-76.
- McCarty, L.T. (1989). A Language for Legal Discourse, I. Basic Features, Proceedings of the Second International Conference on Artificial Intelligence and Law, pp.180-189, Vancouver, Canada.
- McDermott, J. (1988). Preliminary Steps Toward a Taxonomy of Problem-Solving Methods, Automating Knowledge Acquisition for Expert Systems, S. Marcus (ed.), pp.225-256, Kluwer Academic Publishers, Dordrecht, the Netherlands.
- Miller, G.A., R. Beckwith, C. Fellbaum, D. Gross, and K. Miller (1993). Introduction to Wordnet: An On-Line Lexical Database, Five Papers on WordNet, CSL Report 43, Cognitive Science Laboratory, Princeton University, USA, July 1990, revised March 1993.
- Minsky, M. (1974). A framework for representing knowledge, Artificial intelligence memo, No. 306, Massachusetts Institute of Technology, AI Laboratory; also appeared in: Readings in Knowledge
- Representation, pp.245-262. R.J. Brachman, and H.J. Levesque (eds.), Morgan Kaufmann, San Mateo, California, United States.
- Newell, A., and H.A. Simon (1972). Human Problem Solving, Englewood Cliffs, NJ, Prentice-Hall, Princeton, United States.
- Quast, J.A., H.J. van den Herik, and L. Aarts (1996). A Generic Model for the Interpretation of Vague Norms, Legal Knowledge Based Systems; Foundations of Legal Knowledge Systems (JURIX `96), pp.39-46.

- Schreiber, A.Th. (1992). Pragmatics of the Knowledge Level. PhD. Thesis, University of Amsterdam, Amsterdam, the Netherlands.
- Stamper, R.K. (1991). The Role of Semantics in Legal Expert Systems and Legal Reasoning, Ratio Juris, Vol. 4, No. 2, pp.219-244.
- Steels, L.L. (1990). Components of expertise, AI Magazine (summer issue), also appeared as: AI Memo 88-16, AI Lab, Free University of Brussels, Brussels, Belgium.
- Valente, A. (1995). Legal Knowledge Engineering; A Modelling Approach, Doctoral Thesis, University of Amsterdam, Amsterdam, the Netherlands, IOS Press, Ohmsa, Amsterdam, Tokio.
- Visser, P.R.S., and R.W. van Kralingen (1991). Reasoning About Definitions in Statutes, Legal Knowledge Based Systems, Model-Based Legal Reasoning, J.A. Breuker, R.V. de Mulder, and J.C. Hage (eds.) pp.113-122, Koninklijke Vermande B.V., Lelystad, The Netherlands.
- Visser, P.R.S. (1995). Knowledge Specification for Multiple Legal Tasks; A Case Study of the Interaction Problem in the Legal Domain, Computer / Law Series, No. 17, Kluwer Law International, The Hague, The Netherlands.
- Visser, P.R.S., R.W. van Kralingen and T.J.M. Bench-Capon (1997), A Method for the Development of Legal Knowledge Systems, Proceedings of the Sixth International Conference on Artificial Intelligence and Law (ICAIL '97), Melbourne, Australia.
- Visser, P.R.S. and R.G.F. Winkels (1997). Proceedings of the First International Workshop on Legal Ontologies; LEGONT '97, University of Melbourne, Law School, Melbourne, Victoria, Australia.
- Visser, P.R.S., and T.J.M. Bench-Capon (1998). A Comparison of Four Legal Ontologies for the Design of Legal Knowledge Systems, Artificial Intelligence and Law (accepted for publication), pp.1-31.

Notes:

1 WordNet is a commonly used ontology (Miller et al, 1993), which is on-line available at the following URL: http://www.cogsci.princeton.edu/~wn/obtain/

2 The term `Law' itself has 7 different senses.

3 Although the phases are largely executed in the order listed here the creation of an LKBS will involve several iterations through all phases.

4 The use of the word `method' here concerns the method of the LKBS and should not be confused with the system-design method presented in this article.