How to validate a legal knowledge-based system

Schmidt, A.H.J.; van den Herik, H.J.

Published in:
Legal knowledge based systems: An interview of criteria for validation and practical use: Exploring the quality of applications resulting from research programs in the Netherlands

Publication date:
1990

HOW TO VALIDATE A LEGAL KNOWLEDGE BASED SYSTEM

Aernout Schmidt, Jaap van den Herik
Research Group on Expert Systems and Social Security
Leiden University
PO BOX 9520, 2300 RA Leiden, the Netherlands

Summary

The investigation of appropriate validation criteria for a legal knowledge based system entails the identification of four validation problems. Two of them are closely related to well known AI issues: the frame problem and the closed world assumption. The two others, selection and coding, are well known hard problems in social sciences. Three criteria for empirical validation of the system are subsequently developed. Finally, application of the criteria on parts of the system is reported and implications are discussed.

I. Introduction

The state of the art in legal AI does momentarily not allow a strict formulation of general validation criteria for legal knowledge based systems (LKBSs). Accordingly, the general question "which criteria are suitable for the validation of LKBSs ?" remains open. However, having developed a LKBS for probation, in this contribution we deal with the more specific question "which criteria can be supplied to validate this particular LKBS?"

The LKBS referred to is named "Pallas ex Machina" (PEM). It has been built by one of the authors (AS) in 1987 and it has been implemented in SD-Prolog, requiring the standard hardware facilities of a personal computer. The system is meant to represent the development of Dutch legal knowledge on probation between 1971 and 1979. Next to the case knowledge on probation, a considerable amount of peripheral empirical data has been collected and is available for further analysis. Hence the systems seems a suitable vehicle for validation research, especially because the operation and the actual use have been published extensively [Sch87]. Moreover, the research can be concluded since it is unlikely that it will be confused by political issues in the future. The Dutch probation Act has been radically changed in 1986; probation is now formally given by default.

This presentation will follow the various methods of knowledge representation in PEM. In Section 2, we start with a brief description of the system, the representational techniques used and the validation problems they entail. Section 3 summarizes some candidate validation techniques and relates them to the noted validation problems. In the conclusion of Section 4 we apply some of the validation techniques to some of the validation problems as they are instantiated in PEM.

II. The probational system PEM

During the period 1971-1979 probation in the Netherlands was granted or denied by the Secretary of Justice. In this period approximately 1000 decisions were taken

each year. The decisions implied either release on parole or further imprisonment. A decision had to be taken after two thirds (and at least nine months) of the executed prison sentence. Until 1975 the decisions were not open to appeal. From 1976 onwards an appeal court was effective.

The Secretary of Justice based his decisions on paper information, forwarded to him by a standard administrative procedure. The information has been recorded in file maps, accessible for research on special conditions; 426 have been selected, analyzed and coded.

For our validation research, we assume PEM's prospective user to be a legal expert who is given the job to make probation decisions in a legally valid way. The system is considered to support him in this task.

2.1. General aim of the system

Since PEM is primarily a research tool, the general aim of the system is to investigate to what extent an adequate formalization of legal knowledge in a specific domain is achievable. Allowing legal knowledge to be *eo ipse* neither complete nor static nor deterministic, the main result of this research was that an adequate formalization beyond explicit and at least partially accepted legal theory is impossible. Of course, this result has consequences for the validation issue.

We do not assume that legal theory has been always available. When, for instance, the Dutch Secretary of Justice was faced with the first probation decision in 1890, he was (by statute) given complete freedom to decide either way. Hence, the first decision was arbitrary (i.e., there was no legal theory to support the decision). The first decision, however, was added to legal theory, making the next decision less arbitrary. Clearly, allowing space for new decisions creates a validation problem: how should we validate a legal decision if we have no legal theory to support the decision?

So far, we have implicitly assumed that any new decision by the Secretary of Justice takes previous decisions into account. Still, allowing for fully arbitrary decisions obviously disqualifies the idea of building a consistent legal system. In logical terms the Secretary of Justice is able to violate the closed world assumption for the legal domain. In the extreme this implies that the legal world is logically open, which in turn poses a real problem to validation. We will refer to this validation problem as the **jurisprudential opened world assumption problem** (abbreviated as the jurisprudential OWA problem).

A further assumption deserves some attention. When designing PEM, a choice had to be made whether Dutch legal theory was supposed to be variant or invariant. Case law is considered to be variant, all other explicit and generally accepted legal theory invariant. We started with providing PEM invariant legal theory only - of course as much as possible.

Thereafter, PEM was supplied with variant legal theory according to the following procedure. The coded file maps were processed in chronological order. For every file map PEM is prompted to make a motivated decision. When explicit theory is lacking, the system requests a decision (and a motivation) from the user, who is supposed to be a legal expert (the user may input the Secretary of Justice's decision). *PEM adds this information to its variant legal theory.* The procedure is

based on legal (meta) theory, including the principle of *stare decisis*. However, it generates questions as to how case law (as a part of legal theory) is kept consistent, if at all, when the principle of stare decisis is not to be applied rigorously, as is the case in Dutch legal theory.

These questions raise a problem regarding validation of incremental case law representation. The problem is very closely related to the "frame problem" and we will further refer to it as the **jurisprudential frame problem.**

The validation issues arising when PEM's decisions and the decisions actually taken by the Secretary of Justice mismatch will be discussed in the Sections 3 and 4.

2.2. Knowledge sources and representations used

So far the expert user, file maps and legal theory (including case law) have been distinguished as knowledge sources. Generally speaking, file maps have been represented as records (or as Prolog facts) and legal theory has been represented as rules (or as Prolog predicates). We provide more detail below.

2.2.1. File maps

File maps contain *ante hoc* information, classified into two categories: case and context.
Case information reflects the instantiation of notions which the prisoner can influence and for which he can be held responsible. PEM is able to recognize three of these notions: (1) the prisoner's attitude to parole, (2) the probability of recidivism and (3) the readiness to reform. All three notions were coded in three-valued variables, using invariant legal knowledge to transform the information in the file maps. The coding procedure used [Sch87] is assailable and should be validated. This validation problem is introduced as the **coding problem.**

Context information reflects the instantiation of notions the Secretary of Justice can influence and for which he can, accordingly, be held responsible. Context information contains notions regarding (1) special "open" detention, (2) another prosecution pending, (3) pardon, (4) expulsion as unwanted foreigner and (5) the order to be detained during Her Majesty's pleasure (which, at least in the Netherlands, can be inflicted cumulatively to imprisonment). Since we are dealing with dichotomies based on explicitly stated administrative decisions the coding problem is of little urgency for context information.

The consideration for the distinction between case and context lies in the assumption that cases can be grouped within contexts but not across contexts, thus allowing for different legal policies in different contexts. We may expect for instance that parole policy regarding partially pardoned prisoners is different from parole policy regarding the unpardoned. The validation issue involved is the selection and identification of context notions. If too few notions are selected, it may lead to errors invoked by overgeneralization, too many may result in inconsistency with legal common sense. We refer to this problem as the **selection problem.** (Of course the selection problem is not restricted to context information. The problem is of most general scope.)

2.2.2. Legal theory

We have distinguished the invariant legal theory incorporated in PEM into legal methods and legal functions. The legal method performs the task of identifying the legal functions and of prescribing the order in which these functions must operate (on cases, contexts and intermediate results) to yield legally sound inferences.

The legal functions identified are:

(1) proper procedure
This function imports a coded file map and checks the minimum amount and consistency of information which should be available before a decision may be ventured at all (e.g.: the prisoner's opinions and arguments regarding parole should be available; no decision should be taken when a prisoner has escaped and is still on the run). Moreover, the system checks the expiration of terms (e.g.: the decision should be taken before two thirds of the sentence has actually been executed).

(2) balance of interests
This function evaluates the interests involved in the decision. It differentiates between two types of decisions: Open decisions (for which no precedent is known) and closed decisions (for which at least one precedent is known). Open decisions are arbitrary. Closed decisions are, in accordance with stare decisis, by default determined by precedent.

(3) validation of default decisions
This function allows the system's user (1) to authorize a default decision, (2) to take an arbitrary decision or (3) to override a default decision. In the last two instances the user (who is supposed to be a legal expert) is prompted to give a motivation. If the user wishes to override a default decision the system presents the consequences of this correction on earlier decisions.

(4) precedents and criteria
This function records the decisions in a precedent database. If an arbitrary or an overriding decision has been made, the function keeps the precedent database consistent by "forgetting" cases that would have become anomalous under the new criteria (in practice these cases are assigned with a forget flag). Moreover, the function stores changes in criteria in a criteria database and records the related motivation.

For the description of the operation order, PEM's legal method is represented in a Prolog predicate, to be interpreted in a procedural way:

```
method :- repeat,
  proper_procedure,
  balance_of_interest,
  validation,
  precedents_and_criteria,
  fail.
```

This represents a perpetual loop. The assumption is, that this order of functions represents correct jurisprudential procedure.

2.2.3. Precedents and criteria databases

Precedents are represented in records containing a maintenance flag, a context, a date, a case, a decision, PEM's inference and a case number. The records are stored in a database like this:

```
precedent(a,11112,771221,313,p,p,312).
precedent(a,11112,750327,313,p,p,288).
precedent(a,11112,730406,313,p,p,113).
precedent(o,11112,730405,313,d,d,112).
```

The first precedent reads as follows.
The first parameter indicates an active case (a), not having been made obsolete (o) by the precedents and criteria function. The second parameter contains the notions of context information, with 1 (not applicable) or 2 (applicable). *In casu*: only the order to be detained during Her Majesty's Pleasure is applicable (11112). The third parameter indicates the date of the decision, i.e.: the decision was made on december the 21st in 1977 (771221). The fourth parameter represents the notions of case information. The instance given means that the prisoner does not want to be given parole (313), that the probability of his recidivism is small (313) and that he does not show any readiness to reform (313). In the three-valued logic used for representing case information 1 means a big, 2 means a medium and 3 means a small incentive to give parole. The fifth parameter indicates whether the Secretary of Justice actually did decide to give parole (p) or to deny (d) it. The sixth parameter denotes PEM's advice on giving parole (p or d). Finally, the seventh parameter provides the number of the recorded file map (312).

The second and the third precedent in the listing above are also active. The fourth precedent, however, is obsolete (o). In fact this precedent represents a Secretary of Justice's decision taken one day before the third precedent. Although case and context information are identical, a revision of decision occurs. Here we touch upon the main topic of our research. Forementioned revision must be recorded in a criteria database.

In analogy to precedents, criteria are also represented in records (context, date, parole criterion, detention criterion) and are collected in a database. The records stored in this database are as follows:

```
criteria(11112,730406,321,333).
criteria(11112,710101,112,333).
```

Criteria are represented as three-digit numbers in the third and fourth parameters of the criteria records. These numbers must be interpreted much in the same manner as case information in the precedent database. In the precedent database the number denotes descriptive case information. In the criteria database, however, the numbers represent the lower and upper bounds of the space for arbitrary decisions ("das freies Ermessen").

The criteria are used by the balance of interest function to separate arbitrary from closed decisions. Subsequently, the criteria are used to compute default decisions for closed cases. For instance, the first record of the criteria database listed above is used as follows:

- when the case information of a precedent, interpreted as a number, is higher then or equal to 321 and lower than 333 the decision is arbitrary;
- when the case information of a precedent, interpreted as a number, is higher then or equal to 333 then parole will be denied by default;
- when the case information of a precedent, interpreted as a number, is lower then 321 then parole will be given by default.

The additional numerical interpretation of case information presupposes the possibility of (a) the ordering of case notions in accordance with their (global) weights in the probation decision policy and (b) the ordering of codes within case-notion variables in accordance with their (special) weights in the probation decision policy. These assumptions generate coding constraints which will further aggravate the selection and the coding problems.

III. Validation of PEM

The validation of computer programs can be distinguished into (a) formal validation, (b) empirical validation and (c) adequacy assessment.

Formal validation relates formal specifications to programs. The search for formal validation techniques has yielded structured programming [Dah72], proofs of correctness (verification) [Gri81] and logic programming [Kow79, Llo84]. Of course, checking of typographical errors is a prerequisite for the above mentioned validation techniques. In our opinion formal validation applies to programs in every domain. There is no reason to expect specific formal validation criteria for LBKSs.

Empirical validation relates implemented theory (or program behaviour) of an actual world to observable actual world behaviour. If the theory is descriptive, program behaviour can be disqualified when it is not identical with relevant observations. If the theory is prescriptive, observations can be disqualified by program behaviour. A problem arises, when we wish to validate a descriptive theory, because we have to accept the observed behaviour of the world. Contrary, when we wish to validate the actual world behaviour, we have to adopt the prescriptive theory. The quandary can be approached in several ways resulting in a range of validation criteria, among others proposed by De Groot [Gro61]. PEM is considered to be an implemented prescriptive theory on the behaviour of the Secretary of Justice while deciding on probation. From this perspective, we shall treat the validation criteria.

Adequacy assessment examines the aims of an implemented instrumental theory with regard to the observed effects of its application. The criteria may contain political, economical and ergonomic issues, as well as other items of effectiveness. Many of the criteria proposed in [Dek88] and in [Pie89] are adequacy assessment criteria. Adequacy assessment is basically a *black-box* method, applicable to general KBSs and certainly not unique to LBKSs, let alone to a particular LBKS.

3.1. Validation techniques

Considering PEM as an implemented prescriptive theory of the Secretary of Justice's behaviour in his decisions on probation, PEM's inferences and motivations should provide clues in the validation process. These clues should be collected and

processed by an implemented methodology. Accordingly, a set of heuristics is implemented in PEM elicitating legally sound knowledge from an expert user. PEM applies this knowledge for its inferences and its motivations. Subsequently PEM is expected to produce legally sound results. Like other methodologies (e.g., De Groot's empirical methodology [Gro61] and Breuker's and Wielinga's KADS [Bre87]) PEM is not deterministic. For the quality of its domain knowledge it depends heavily on the expert user. Nevertheless, the question remains: how do we validate PEM?

One of the most important problems in legal AI deals with the assumptions made before developing a LKBS (e.g., [Lei86, Sus86]. Hence a preliminary question is in order:

Assume that legal experts reach a decision using legal sources on which a legal method is applied; assume further that the legal sources and the legal method can be formally represented (and assume for the moment that they are actually implemented in a LKBS). The question now reads: are the inferences of that particular LKBS legally valid?

The tentative answer to this question is affirmative, otherwise legal AI had lost part of its research ground. However, it is an open question whether the application of validation criteria makes this answer more convincing. Moreover, a further assumption should be made before we can start on empirical validation. The plain question is:

Assume the legal method to be deterministic, is it then possible that inferences of a legally valid LKBS are validated empirically?

Obviously, not even for a simple system such as PEM both questions mentioned above can be answered with authority. Too many validation problems (especially the jurisprudential OWA problem and the jurisprudential frame problem) are unsolved.

What remains is the investigation to what extent the system's behaviour can be validated (i.e. we may validate parts of the system in isolation) and to what extent empirical validation assures us the legitimacy of our research. Below we provide a brief overview of some validation techniques first. Subsequently we reexamine the validation problems mentioned earlier.

3.1.1. Empirical validation: appraisal by expert forum

A clear candidate technique for the validation of the system's behaviour is the well-founded legal appraisal of PEM's results by a forum of experts. This type of validation is of great merit for an overall validation and may also be useful for the appraisal of the behaviour of isolated functions.

3.1.2. Empirical validation: experiment

Another clear candidate technique for the validation of parts of the system's behaviour is empirical experiment. We give a brief illustration.

One of the basic assumptions incorporated in PEM is the arbitrary decision. If we assume that the legal method on parole is deterministic, and if we further assume

that arbitrary decisions are made indeterministically, then the conclusion must be that arbitrary decisions come from outside the legal method. The existence of arbitrary decisions may now be tested by experiment. Judges of the appeal court and administrators of the Secretary of Justice can be requested to give decisions on arbitrary and on closed cases. Our hypothesis would be that their decisions will vary among arbitrary cases and that they will coincide on closed cases.

3.1.3. Empirical validation: policy prediction

Although PEM is meant to validate the Secretary of Justice's decisions, we also can use his decisions as predictions, which may later be completed with corrections. The method of prediction and correction has been adopted from numerical analysis. The type of validation is based on the assumption that the Secretary of Justice will use a generally accepted legal method most of the time. It will give a clear indication of invalidity if the Secretary's decisions are disproportionally malpredicted.

3.2. Validation problems, suggested criteria

The jurisprudential OWA problem presupposes that probation decisions (and indeed legal decisions in general) can be classified either as open (arbitrary) or as closed. Closed decisions can be validated because legal theory is available. However, open decisions can not be legally validated. As a corollary the jurisprudential OWA problem yields an overall criterion for the validation of LKBSs. Criterion (1):

A LKBS should contain a decision procedure for the classification of open and closed decisions.

Such a decision procedure is in itself subjected to validation using the techniques mentioned above.

Allowing legal world knowledge to include case law entails the *jurisprudential frame problem*. Its core is that case law may be inconsistent over time. Hence, case law should be managed somehow. In the Dutch legal theory we have been faced with a hard problem, because high courts do not actually apply the stare-decisis principle rigorously and are not even expected to do so. This seems to imply that for the high courts any decision is arbitrary, but this would lead to inconsistency with legal common sense. The main point here is how to manage what is called the 'overriding decisions'. Although there does not exist any generally accepted applicable rule of law, at least some relevant heuristics should "protect" the consistency of the entire legal system (hence emulations in the formal LKBSs should be consistent). If not, both systems would be immune to validation. We cannot accept that. As a corollary we state criterion (2):

A LKBS should contain heuristics for keeping its precedent database consistent.

Again, such heuristics may be subject to empirical validation.

The *coding problem* refers to the possibility that the code used does not correctly reflect the information meant to be represented. This problem has been extensively described elsewhere and criteria (such as repetition of the coding process by different people) can be found in [Gro61]. The coding problem is one of the most difficult to overcome; it may confuse the empirical validation of a system's overall

behaviour. Practice has taught us that indications for coding errors may be found by applying policy prediction techniques to overall system behaviour. Where malpredictions of the system are incidental and inexplicable, coding errors may prove to be the cause. This can only be established by reexamining the original file maps.

The *selection problem* regards questions such as how many different types of variables should be used and how many categories per variable are needed for coding. This is a well-known problem in information systems as in the database world. The problem imposes requirements on an administration. The granularity and the selection of information and its management is rather complicated and deserve to be treated separately.

Hence we do not propose any direct validation criteria for the coding and selection problems; we only identify a prerequisite for finding them. Criterion (3):

A LKBS should register its own behaviour, with emphasis on the maintenance of precedent and criteria databases.

IV. The criteria applied to PEM

In this Section we discuss the three criteria introduced above, applied to PEM. In a brief summary we may state that PEM has a decision procedure for open and close decisions, it contains heuristics for the management of its precedent database and it contains a registration of its own behaviour in precedent and criteria databases.

4.1. The decision procedure for open decisions

The criterion rests on the assumption that in Dutch probation open decisions do actually exist. The assumption can be validated empirically be presenting open and closed cases to legal experts, requesting them to decide upon them and to motivate their decisions. *Provided that we do not allow arbitrary decisions to occur*, we are forced to conclude that Dutch legal theory is either indeterministic or inconsistent, if prominent experts' decisions contradict. We note in passing that indeterminacy and inconsistency can lead to the same result, due to the dichotomy chosen.

Contrary, *if we allow arbitrary decisions to occur*, we must assume that these are based on theory which is non-legal. Many of such theories can be thought of. We even can expect to encounter a Gaussian distribution of arbitrary decisions. This being so, we predict that the expert decisions on open cases will vary.

Of course, closed decisions should be decided upon unanimously.

In 1980 Schmidt has interviewed seven judges of the appeal court and eight probation executives of the Secretary of Justice. He has asked them to decide on two open cases and on one closed case. The judges and executives showed full preparedness to cooperate, with one exception only. One judge refused to give an opinion on the second open case.

A brief description of the cases follows:

case (1) An ex-prisoner on parole wishes his detention to be resumed and asks for the withdrawal of his probation. He states to commit a crime if the withdrawal is denied. The question now is: would you withdraw his probation? (This case is considered to be open since no precedent was known).

case (2) A prisoner will subsequently be detained during Her Majesty's pleasure, as soon as probation is granted. However, he abhors detainment and threatens to kill himself if probation is actually granted. The question is: would you grant probation, knowing that the detainment is to follow anyway? (This case was considered to be open since only one precedent was known which could not be related to a statement of policy).

case (3) A prisoner has been convicted because he battered his (ex) wife. Recently, she received postcards of a threatening nature. No connection between the prisoner and these cards can as yet be proven. The question now is: would you grant parole to the prisoner? (This case was considered to be closed since many decisions have been made on similar cases. The relevant policy so far was: granting probation, unless a prosecution had formally been initiated).

In figure 1 the results of the experiment are presented. As predicted, the judgements of the fifteen experts varied widely on the open cases, whereas the closed case was unanimously decided upon by the seven appeal court judges. The eight probation executives apparently did not consider this a closed case.

	Case	Parole	Detention
Probation executives (8)	1	5	3
	2	3	5
	3	6	2
Appeal court judges (7)	1	4	3
	2	3	3
	3	7	0
Total		28	16

Figure 1 An experiment on open and closed cases

The experiment settles the issue of open and closed cases in no way definitely. Neither does it provide an outright refutation of our hypotheses.

4.2. Heuristics for keeping the precedent database consistent

PEM's heuristics for keeping the precedent database consistent are straightforward: (1). Decisions are made in chronological order;

(2). Whenever a criterion is changed, all precedents anomalous with the new criterion are made obsolete;
(3). Whenever the user wishes to make an overriding decision he can do so after explicitly changing the appropriate criterion, and after providing a motivation.

Prima facie the heuristics work rather well, but they have not been validated. The main validation problem mentioned above (cf Section 3.2.) regarding overriding decisions proved actually quite restricted due to the motivation heuristic. This heuristic shows natural to many legal experts. Since this is no foundation for statements of general validity further research in this area is in progress in Leiden.

4.3. The registration of PEM's behaviour

PEM is made to register its own behaviour (and the behaviour of its user) in precedent and criteria databases. Validation of actual behaviour by policy prediction gives an indication of the selection problem being urgent in one specific context. PEM's behaviour deviated systematically from the Secretary of Justice's in the context where another (i.e. a new) prosecution is pending when a probation decision is due. We think that in this context so little information was available in (and hence selected from) the file maps that PEM considered most of the decisions to be closed, whereas the Secretary of Justice judged them actually to be open. The set of cases which PEM considers to be closed and which the Secretary of Justice considers to be open provides a difficult validation issue. Either the Secretary of Justice's behaviour is in conflict with legal theory, or he has more information available than PEM. Since there are very few sets of cases triggering this issue, we have a startingpoint for further investigation. With regard to PEM this investigation could not be effectuated, due to nonexistent or inaccessible written case information in the uncoded file maps.

V. Conclusions

We have distinguished three types of validation: formal validation, empirical validation and adequacy assessment.

In passing we note again that accepted criteria for formal validation are directly applicable to LBKSs (cf Section 3).

Unfortunately, empirical validation criteria are hard to find, even for one specific LKBS. The main reasons seem to the validation problems mentioned in Section 2.1, the jurisprudential opened world assumption problem and the jurisprudential frame problem. However, the scope of these validation problems might reach far beyond the LKBS in question.

Adequacy assessment, as a black box method relating aims with effects, yields no special type of criteria for LKBSs.

In summary, the validation criteria we suggested in Section 3.2 are of a general nature and represent prerequisites for finding more specific criteria. In the application of our criteria to (parts of) PEM we showed that the search for more specific criteria may use empirical methodologies.

VI. Acknowledgements

The authors are grateful for support from many sides, although we mention but a few. We recognize the help of the Secretary of Justice (making file maps available), of judges of the Dutch probation appeal court and the administrative executives of the Justice Department (showing preparedness to answer questionnaires) and of IBM Nederland NV (sharing resources with the authors in a joint research project on a prototype of a LKBS for aspects of social security).

VII. References

[Bre87] Breuker, J.A., and B.J. Wielinga (1987) *Use of models in the interpretation of verbal data*. In Kidd (ed.), Knowledge acquisition for expert systems: a practical handbook, New York.

[Dah72] Dahl, O.-J., E.W. Dijkstra and C.A.R. Hoare (1972) *Structured Programming*, London.

[Dek88] Dekker, S.T., J. Henseler and H.J. van den Herik (1988) *Kwaliteit van Expertsystemen*, Proceedings AI Toepassingen 1988, NGI-SIC 1988.

[Gri81] Gries, D. (1981) *The science of Programming*, Berlin.

[Gro61] Groot, A.D. de (1961) *Methodologie, Grondslagen van onderzoek en denken in de gedragswetenschappen*, den Haag.

[Kow79] Kowalski, R. (1979) *Algorithm = logic + control*, Communications of the ACM 22, pp 424-436.

[Lei86] Leith, P. (1986) *Fundamental errors in Legal Logic Programming*, The computer journal, 29, 6, 545-551.

[Llo84] Lloyd, J.W. (1984) *Foundations of Logic Programming*, Berlin.

[Pie89] Piepers, P.A.W. (ed.) (1989) *Toogdagbundel juridische informatiesystemen*, NVIR 1989.

[Sch87] Schmidt, A.H.J. (1987) *Pallas ex Machina*, Lelystad.

[Sus86] Susskind, R. E. (1986) *Expert systems in law: a jurisprudential inquiry*. Ph. D. thesis, University of Oxford.