Legal knowledge based systems
JURIX 92
Information Technology and Law

The Foundation for Legal Knowledge Systems
Editors:
C.A.F.M. Grütters
J.A.P.J. Breuker

H.J. Van den Herik
A.H.J. Schmidt
C.N.J. De Vey Mestdagh

N. den Haan, TRACS: a Support Tool for Drafting and Testing Law, in: C.A.F.M. Grütters, J.A.P.J. Breuker, H.J. Van den Herik, A.H.J. Schmidt, C.N.J. De Vey Mestdagh (eds.), Legal knowledge based systems JURIX 92: Information Technology and Law , The Foundation for Legal Knowledge Systems, Lelystad: Koninklijke Vermande, pp. 63-70, 1994 ISBN 90 5458 031 3.

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

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

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

TRACS: A SUPPORT TOOL FOR DRAFTING AND TESTING LAW

N. DEN HAAN
Department of Computer Science and Law, University of Amsterdam, Amsterdam, The Netherlands

Summary

This paper discusses the results of the implementation of a prototype system for the application of traffic law. The problems tackled in this system are the computational complexity of legal reasoning regarding the use of knowledge about the world, and the representation and reasoning with deontic modalities. Testing two paragraphs already rendered several important notes about the design of the traffic law. This paper reveals the experiences and results, and gives directions for further research.

1. Introduction

In the winter of 1991 the TRACS project was started. It was supported by a grant of the Dutch Association for Scientific Research into Traffic Safety (SWOV). Initially, the project aimed at the construction of an intelligent teaching system (ITS) for law. However, since the SWOV is also concerned with testing new versions of the traffic law as presented to them by the Ministry of Transport and Public Works, the application of law became the core interest. This explains the TRACS acronym: Traffic Regulation Application and Comparison System. Applying law is the basis both for legal ITS and for testing law. The design of the system was presented in [Den Haan & Breuker, 1991]. The prototype, which was programmed to perform the elementary legal-reasoning processes involved in law application, proved to have important results.

The actual application of law is the core activity of legal reasoning. An other important issue is the analysis of facts and the interpretation of vague terms. Not being the goal of our research, they have not been embedded in the system. The TRACS project aims at regulation-based reasoning and embodies the application of the traffic law. The legimitization of the results is considered to be a task that, in our opinion, requires thorough legal experience and up to date knowledge of the social structure.

The implementation of the system has proceeded as planned in the presented design. The next section will recapitulate this design, in order to provide a framework for the discussion of the results. Section 3 reflects on the representation formalism used in the prototype. Subsequently, section 4 elaborates on the tests, and section 5 shows what implications for law drafting have surfaced.

2. TRACS design

The short term goal for the functionality of TRACS is the application of law. The legal reasoning of TRACS involves the search for applicable rules and the selection of rules with the highest priority. The three layers of legal reasoning (application, selection and validation) are clearly visible in figure 1. In short, rules are found applicable if facts in the situation description can be matched to the grounds for rule application, as stated in the condition parts of rules. Priority rules are used to determine an order in the set of applicable rules. For each law text, specific priority rules can be found, sometimes stated in the text itself. Additional general priority rules are defined for the entire law domain. The validation process is quite similar to the application process, but matches the conclusions of rules to the situation description.

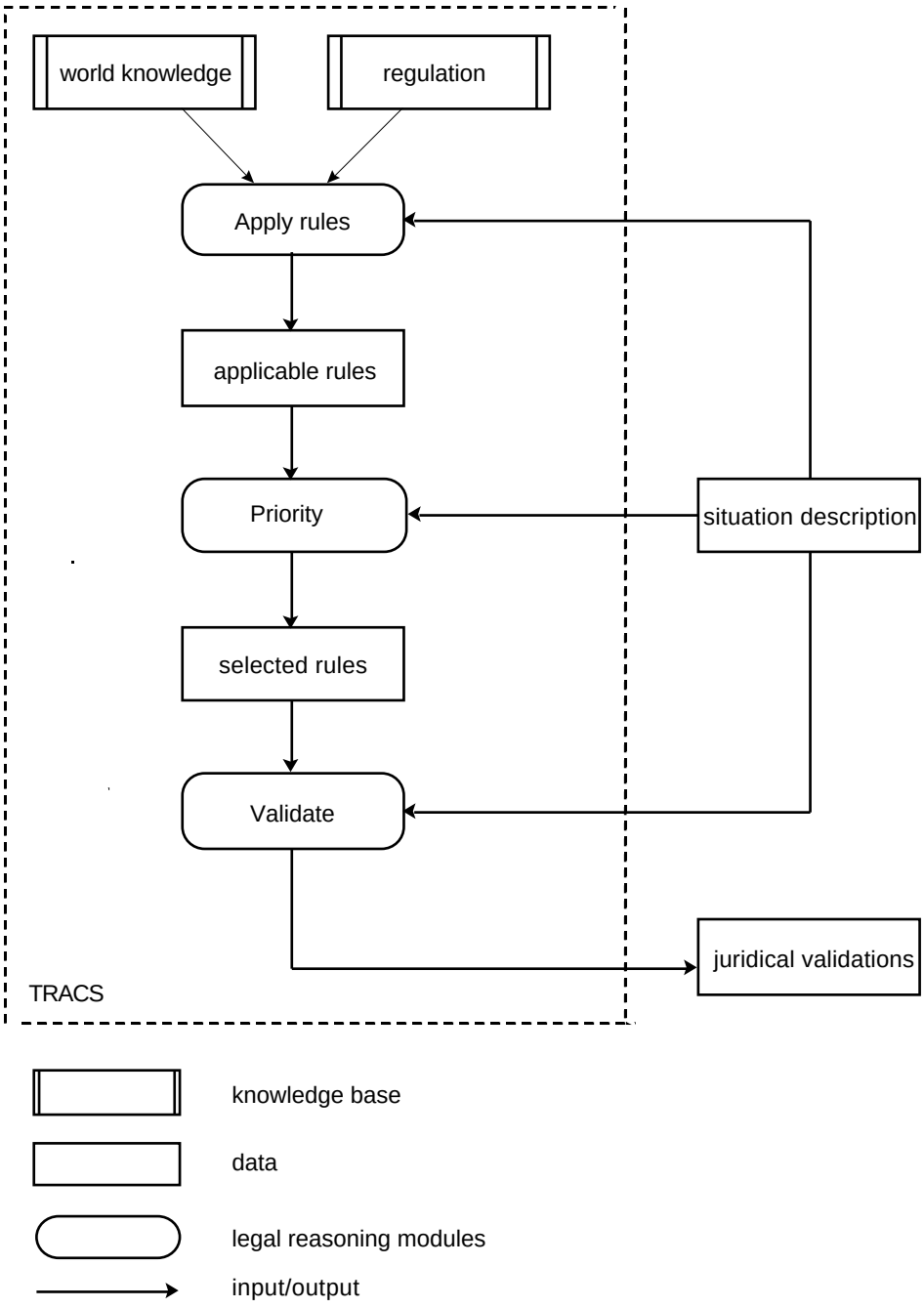

Figure 1: TRACS architecture

The rules in the regulation knowledge base (KB) refer to knowledge which is not explained in the articles themselves. To overcome ambiguity of the most important concepts, a definition list is given in the beginning of each law. This list defines some concepts, but does not give a complete understanding of the world. To solve this problem, a separate knowledge base has been set up in which all the world knowledge concerned here is recorded. Not only are now complex concept descriptions in the rules unnecessary, but also the legal reasoning processes may refer to the world knowledge. The world knowledge is not scattered anymore over numerous articles, but is collected in the world KB. In this way the modeling of the traffic world is better controlled [Bench-Capon, 1989]. In order to be interpreted, each world knowledge term from the law text has to be correlated to other pieces in the world KB. Type hierarchies define subtype and supertype relations between the concept types of the traffic world. There are three main

type hierarchies, one for traffic participants, one for traffic actions and one for road elements. Their main use is to support the interpretation of concepts in the articles. Matching types of situation facts to concept types in the articles resembles the sorted deduction principle of Frisch [1991]. Terms do not match when the concept type is higher in the type hierarchy, or in a different hierarchy.

The requirements for the knowledge-representation formalism were that the representation of the traffic law should be isomorphic with the text of the traffic law. This has numerous advantages, e.g., it is easy to correlate test results to the articles or changes in the articles. In previous law representations [e.g., Sergot et al., 1986] this isomorphism had not been respected, resulting in rules that were hard to read, because the rules had to embed interpretation guidelines for complex concepts. The next section deals with the use of the proposed knowledge-representation formalism.

3. Observations on the representation formalism

To assess the juridical validity of a situation, the system compares it to the juridically ideal world, as it is prescribed in the law. When the idea of a juridically ideal world is used in the legal reasoning modules, the representation formalism is no longer in need of modal operators. Removing the modal operators requires that only one modality is used. Since law is seen as a juridically ideal world, and it is obliged to obey the law, the law text had to be O-based, i.e. only contain obligations. The modality that expresses prohibitions can be rewritten in terms of an obligation, because $F(p) = O(\sim p)$ [Hilpinen, 1981], since to forbid an action is the same as obliging that an action is not performed. Brouwer [1990] also discusses negations in regulations but needs extra deontic operators to take care of internal negations and rewrites permissions to conditional exceptions in general rules. In short, permissions are not easy to deal with. According to deontic theories, they are rewritten to $P(p) = \sim O(\sim p)$, i.e. it is not obligatory that not *p*. The obligation modality could not be moved out, because it is not intended that the first negation could eliminate the second negation. The roles of the two negations are different: the first is a modal negation, and the second a normal negation. The inference mechanism provided a solution. Suppose the two negations could be eliminated, the permission is rewritten according to the last rewrite rule from $P(p)$ via $\sim O(\sim p)$ and $\sim\sim p$ to *p*. This would mean that *p* is now obligatory. However, the condition sets are tested in order to determine the applicability of rules, so in these sets the scopes of the rules are set. Permissions mean that when *G* occurs, the situation is validated positively, but when *G* is not true or does not occur, the situation is valid as well. By adding *G* to the condition set this very effect has been established. The rule fires when *G* is the case, and obviously is validated positively, but the rule does not fire anymore when *G* does not occur.

The following example shows how the three types of modalities are rewritten:

Consider the following set of rules:

1	A	->	O(G)
2	A & B	->	F(G)
3	A & B & C	->	P(G)

They would be represented as:

number	conditions	conclusions
1	A	G
2	A & B	~G
3	A & B & C & G	G

Figure 2: Preliminary representation scheme

The situations below should be evaluated as follows: When searching for *applicable* rules, the left hand sides (conditions) of all rules are compared with the situation. To determine the *priority* , the instantiated rules and meta-rules are used. For the *evaluation* the right hand side (conclusions) of the selected rule is compared with the situation.

Situation	Applicable	Priority	Evaluation
A	1	1	not ok
A, G	1	1	ok
A, B	1, 2	2	ok
A, B, G	1, 2	2	not ok
A, B, C	1, 2	2	ok
A, B, C, G	1, 2, 3	3	ok

Table 1: Evaluations

Despite the fact that the terms "conditions" and "conclusions" are used in this representation formalism, the left and right hand sides of the rules are not connected by logical implication. In the representation scheme, the rules are represented in a table. The conditions of rules provide their grounds for applicability, and their conclusion sets indicate what actions are subsequently obligatory, forbidden or permitted. This means that the rules themselves are not truth-validated, but only the terms in the condition an conclusion sets are validated. The method used in the prototype turned out to be crucial for the representation scheme. By validating terms only, the problems with deontic modalities were largely met. The translation of the permission must look very awkward: it is completely tautological. The disadvantage of this formalism is that it is not possible to use automatic proof procedures. The representation formalism, especially the solution for the representation of permissions, is dependent on the design of the legal reasoning modules. At the moment approaches are being developed to prepare for a method in which rules and meta-rules can be validated in automatic proof procedures.

4 . Experiences and results

The task of the prototype is to test specific traffic situations in terms of two paragraphs from the RVV-90. These paragraphs concern the positions of traffic participants on the road (cf paragraph 1) and how they should give way at crossings (cf paragraph 5). Initially, we started with the ordering of the articles as the meta-rule that establishes priority between applicable articles. The designers of the traffic law claimed that the new traffic law is ordered on specificity, and proposed the use of this meta-rule [Den Haan, 1991]. This immediately gave rise to extremely large exception structures: whenever agents complied with a high-numbered article, all the articles with lower numbers could be breached without any notice by the system. Due to the numbering of the articles, and the use of the meta-rule, each rule was an exception to all the rules before it. This problem was solved by narrowing down the scope of this meta-rule to paragraphs only. This is justified by the fact that the design of the RVV-90 is highly modular; each paragraph concerns a unique subject. Within these paragraphs the articles are ordered on the specificity of their application. This correction of the meta-rule pruned numerous erroneous results.

The situation in figure 3 was tested by TRACS:

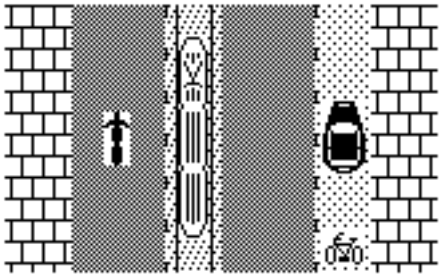

Figure 3: A traffic situation drawn by TRACS

In situation of figure 3, the following outcome was expected:

* The motorbike should be riding on the right side of the road; it is breaching article 3.1.
* The tram is obeying all traffic rules.
* The car should not use the bicycle lane but a normal driving lane (article 10.1).

Instead, what the program came up with was:

* The motorbike was found to obey the rules. The ordering of rules is the cause for the selection of 10.1.
* The tram has breached article 10.1.
* The car was breaching article 10.1.

Since the motorbike is using a normal driving lane (article 10.1, see Appendix) it is no longer obliged to drive in the rightmost lane (article 3.1). According to the mistake in the evaluation of the motorbike, either the meta-rule is wrong, or article 3.1 is not the most general rule. In paragraph one, article 3 states that all traffic participants should drive in the rightmost position on the road. Articles 4 through 10 define obligatory lanes for specific participants. The problem can be solved by adding statements about driving in a rightmost position to the articles defining road positions for the traffic participants. Another solution would be to define a meta-rule for this paragraph which states that first one should pick the obligatory lane, and drive rightmost in that lane. After some examination, the error about the position of the tram can be explained by incomplete knowledge in the world KB about the structure of roads. This knowledge has been derived from the concept descriptions in the traffic law which now turn out to be ill defined. All the building blocks of the road are defined to be on the same level in the type hierarchy. Instead, the types "bus lane", "tram lane", "bicycle lane" are subtypes of "driving lane".

Traffic experts testing the traffic law, can now try to find out how the problems could be repaired. Then they can try to evaluate the traffic situations again (this iterative process is very similar to what usually takes place between the Ministry of Transport and Public Works, and the SWOV). For instance, when the typing of the lanes is corrected to solve the problem about the trams, cars are now allowed to drive on tram, bus and bicycle lanes since these are subtypes of "driving lane". This effect is also unintended, so the repair should be accompanied by repairs of resulting errors. In this case the prohibition to pass the solid lines (that usually surround bus lanes) may offer some help.

Even in a small testing environment like this, the TRACS prototype was able to discover several errors in the current version of the traffic law. When we will upgrade the system to include all 29 paragraphs of the traffic law, the number of interdependencies will doubtlessly increase. Then TRACS can be used to its full extent.

5. Implications for legislation

Legal KBS are able to *apply* law texts, and are therefore very useful for testing out laws under development. These systems are not misguided by surplus background knowledge about the law and its application. Human expertise is always contaminated with more knowledge than is represented in law texts, so when legal experts test laws for coherence and correctness they might overlook ill-stated articles. When this type of system would be used in testing out newly drafted laws, hidden errors may surface. There are various types of errors and repairs. When the application grounds of a rule (its conditions) are too general, the rule will be applied too often. The fact that the representation is isomorphic with the original law text, and the separate representation of world knowledge help domain experts to unravel these errors. Several categories of changes in the law can easily been administered and tested:

* The contents of each article with respect to its application grounds.
* The concept definitions (these are present in the world knowledge base)
* The interaction of the articles and paragraphs.
  Combinations of rules and paragraphs, in combination with:
* The law text as a whole, with the interactions between the rules, articles, paragraphs, priority rules: i.e., as it should be applied in real life cases.

The layered legal-reasoning process gives domain experts a clear view at the law application mechanism. After the first module, all applicable rules can be examined. When the application grounds are too narrowly scoped, a rule is not selected. Also when the world knowledge is wrong, and the concept type is too specific, a rule is not selected. The reverse holds for too widely scoped rules and too general concept types: unintended rules are found to be applicable. The application of priority rules takes place in the second module, so changes in the meta-rules are very clear, e.g., perhaps the right rule had been selected by the system, but did not get the right priority.

6. Conclusions

An automatizated environment in which the law can actually be tested concerning content is a strong support for law design. This environment encourages creative thinking and problem solving among law drafters. The representation of the full law text and the application of verified legal-reasoning methods ensures that testing proceeds according to the letter of the law. The important advantage is that where human experts can be blind for alternative solutions, a legal knowledge-based system will always find and consider all applicable rules. This means that in every phase of law drafting a quick and complete test can be performed.

When traffic law experts are testing the RVV-90, they construct traffic situations that according to their experience will be problematic. In the prototype, the traffic situations have to be encoded by hand. The only problem with the current prototype of TRACS is, that it is not very easily accessible for the domain experts. Traffic law experts do not want to be bothered with programming details needed to enter new situation descriptions. In the future one might opt for a menu-driven drawing tool, so users can easily draw up test situations. In earlier stages of law drafting, all small details will also have to be tested. Furthermore, it is necessary to test countless "low level" situations because a small set of typical situations is insufficient. The use of a situation generator may be helpful. The generator can enumerate all possible combinations of world knowledge for testing. In the same fashion two versions of a law can be automatically compared. While testing a law, users may want to influence the legal-reasoning processes: one might want to check the situation from the perspective of only one of the participants, or maybe users only want to check the application of certain paragraphs or meta-rules. This type of inspection is very important to control the steps in the legal-reasoning processes.

As a conclusion, based on our findings, we believe that the strongest impact of legal knowledge-based systems may be supporting the law drafting process. However, we would like to remark that TRACS also can be used as a simple planning system. To ascertain the validity of a future situation the system can look for supportive articles and thus deduce which additonal actions have to be performed. In the future, advisory systems are feasible that will perform litigation planning. Legal knowledge-based systems will gain even more importance when the initial processes of legal reasoning, i.e., qualification and analysis, and the final processes, i.e., legimitization and motivation of the results, are supported.

The presented testing environment gives a clear insight in the structure of a law. It offers good facilities for testing of several types of errors. When the ordering of rules was used as a meta-rule, unintended exception structures arose because now each article is an exception to all the articles before. The effects of resulting adjustments can also be examined. While testing just two paragraphs already several repairs have been discovered for the correction of the traffic law. We expect that when the entire traffic law is incorporated in the system, and the priority rules are studied more closely, the system may find numerous flaws. The use of an automated environment for testing new versions of the traffic law may have a strong impact on the length of the law drafting process.

7. References

This article may only be cited as: Haan, N. den, TRACS: A Support Tool for Drafting and Testing Law. In: Grütters, C.A.F.M., J.A.P.J. Breuker, H.J. van den Herik, A.H.J. Schmidt and C.N.J. de Vey Mestdagh (eds), *Legal Knowledge Based Systems: Information Technology & Law, JURIX '92*, Koninklijke Vermande, Lelystad, NL, 1992.

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

[Brouwer, 1990] Brouwer, P.W., *Samenhang in Recht*, PhD thesis, University of Leiden, Wolters Noordhoff, Groningen, 1990.

[Den Haan & Breuker, 1991] Haan, N. den, and J.A.P.J. Breuker, A tractable juridical KBS for teaching and applying traffic rules. In: Breuker, J.A.P.J., R.V. de Mulder and J.C. Hage (eds), *Legal Knowledge Based Systems: Model-based legal reasoning, JURIX'91*, Koninklijke Vermande, Lelystad, NL, 1991.

[Den Haan, 1991] Haan, N. den, Interview met de heer Salomon, Ministerie van Verkeer en Waterstaat, *Technical Report LRI-91*, Universiteit van Amsterdam, November 1991.

[Frisch, 1991] Frisch, A.M., The substitutional framework for sorted deduction: fundamental results on hybrid reasoning. In: *Artificial Intelligence*, vol. 49, nos. 1-3, pp. 161-198, May 1991.

[Hilpinen, 1981] Hilpinen, R. (ed), *Deontic Logic: Introductory and Systematic Reading*, 1981, reprinted, first published in 1971.

[Sergot et al., 1986] Sergot, M.J., F. Sadri, R.A. Kowalski, F. Kriwaczek, P. Hammond and H.T. Cory, The British Nationality Act as a logic program. In: *Communications of the ACM*, vol. 29, no. 5, pp. 370-386, May 1986.

8. Appendix

From chapter 2, paragraph 1 of the Dutch traffic law RVV-90:

Article 3:

1. Vehicles are obliged to keep to the right as far as possible.
2. Cyclists are allowed to ride next to each other.

Representation (details about the representation of permissions and the use of typed variables have been explained in [Den Haan & Breuker, 1991]) :

```
article(2 1//3 1,
     subject(V, vehicle),
     (true -> farmost right(V))).

article(2 1//3 2,
     subject(B, bicycle),
     (given(B2, bicycle) &
      ride next(B, B2)
     ->
      ride next(B, B2))).
```

Article 10:

1. Other drivers than mentioned in the articles 5 through 9 use a driving lane.
2. Drivers, except for cyclists, motorcyclists and drivers of vehicles for disabled persons are not allowed to use cycling lanes with solid lines.

Representation:

```
article(2 1//10 1,
     subject(V, vehicle),
     (~ is a(V, bicycle) &
      ~ is a(V, motorcycle)&
      ~ is a(V, vehicle disabled persons)&
      gegeven(D, driving lane)
     ->
      gebruikt(V, D))).

article(2 1//10 2,
     onderwerp(V, vehicle),
     (~ is a(V, bicycle) &
      ~ is a(V, motorcycle) &
      ~ is a(V, vehicle disabled persons) &
      given(Bl, bicycle lane) &
      part of(solid line, Bl)
     ->
      ~ use(V, Bl))).
```