Legal Knowledge Based Systems

JURIX'96

Foundations of legal knowledge systems

The Foundation for Legal Knowledge Systems

Editors:
R.W. van Kralingen
H.J. van den Herik
J.E.J. Prins
M. Sergot
J. Zeleznikow

Paper version: Tilburg University Press, ISBN 90-361-9657-4, NUGI 699

Contents

CONSTRUCTING NORMATIVE RULES

Nienke den Haan, Joost Breuker
Department of Computer Science and Law, Faculty of Law, University of Amsterdam
P.O. Box 1030, 1000 BA Amsterdam, The Netherlands
email: {nienke, breuker}@lri.jur.uva.nl

Abstract

Regulations can be structured and phrased in a large variety of ways. In this paper procedures for automatic generation of these paraphrases are presented. First the set of all possible situations in a domain is generated on a model of this domain. Next, these situations are qualified as legal or illegal: the qualification model (QM). A rule generator turns this QM into a regulation. By changing parameters different versions of the same set of norms (QM) are obtained. In this paper we describe the algorithms for the rule generator (see also Den Haan and Winkels (1994), Winkels and Den Haan (1995), Den Haan et al. (1996). An example of the process is presented in the last section. We conclude that these tools enable the semi-automatic support of core activities in legislative drafting and point out the benefits of having paraphrased versions of regulations.

1 Introduction

Legislative drafting can be semi-automatically supported in at least three different ways which are complementary. The first one is by providing information services that relate general legal information to the specific tasks of the legal drafters. An example is the LEDA system, that follows the officially recommended drafting procedures, directives and styles Voermans and Verharen (1993). LEDA offers access to relevant legislation in hypertext format, and provides the initial normative structure formats. A second way is by providing tools that check the consequences of a regulation. EXPERTISZE Svensson et al. (1992) and TRACS Den Haan (1996) are examples. In this paper we present a third kind of tool and approach which automizes the construction of paraphrases of regulations on the basis of normative goals ((un)desired social behaviour) and a model of the domain to be regulated. The same normative goals can be expressed in different ways for different purposes (*e.g.*, legal subjects concerned), like paraphrases of text that express the same underlying conceptualization Winkels and Den Haan (1995). In this paper we present procedures for generating paraphrases of norms (besides normative statements (“rules”), a regulation may contain also other types of statements, in particular definitional ones and reactive ones see Valente (1995) for a typology of legal knowledge).

2 An outline and rationale for generating normative rules

The basic assumptions and steps in the process of generating regulations can be summarized as follows. A legal domain refers to a particular world that has to be regulated, where world means some relatively independent subsystem in a society. To describe such a world one needs a terminology or, rather ontology Valente (1995), Valente and Breuker (1996). A description of how such a world actually *works*, *i.e.,* what behaviours can be exhibited by this world is called a world model. Behavioural models can be stronger than that. Causal relations may be identified between situations so that actual *predictions* can be made. In the (largely implicit) models of worlds that underlie law prediction is not a requirement. On the other hand, effective enforcement of the law requires some predictability, *e.g.,* as to how citizens will accept the rationale of a law. Behaviour at a particular moment is called a situation, consisting of some configuration (relations) of objects which are in a particular state. Situations can be instantaneous (actual), or generic (abstract). Normative goals are generic situations marked as being undesirable/illegal or desirable /prescribed (see below).

A world model is the set of all possible situations. However, such an enumeration is not very parsimonious or explicitly coherent. Therefore, world models are constructed as generic abstractions as to how objects may relate to one another and in what states objects may occur. For physical worlds, the descriptions may rely on physical principles, and states may parsimoniously be abstracted into simple qualities (cf. qualitative reasoning). However, for social worlds, the constraints that describe what kind of objects may engage in what relations with what other kinds of objects are far more difficult and complex to specify, in particular because human agents, who are the most central object in these models, are by themselves already utterly complex. Therefore, a first step in drafting a regulation is in modeling a legal world, *e.g.*, income tax, traffic, copyrights. In current practice, no explicit modeling occurs, but it is implicit in the understanding of the problems and the debates - political and technical - that are preliminary to and contingent upon the drafting activities proper. Making such a model explicit, as automation requires, may have the important advantage of clarifying more precisely and coherently what is at stake in a law. It may result in well defined terms, which may be an explicit part of a regulation as well. The disadvantage is also obvious: it takes a lot of effort - up to one or more personyear - which may be relatively large for small legal drafting projects (*e.g.*, local repairs). A second step is going from an articulate and parsimonious world model into a version that consists of a list of all possible situations: this we call situation generation, and can be performed automatically (see section 3).

The partioning of a world model into two disjoint sets of illegal (undesired) and not illegal (not undesired) is called the Qualification Model (QM) (see Den Haan (1996)). Because in (political) practice normative goals (rather: intentions) are formulated in more global and vague terms than legislation requires, the process of turning these into the QM specification involves assessment.

The final step involves the construction of one or more regulation models (RM). An RM is not the same as the text of a regulation. It consists of simple frames, called "rules". These rules have a superficially similar structure as rules in rule based representations, but their interpretation is certainly not the same: in particular because rules in a rule base have no interpretation for deontic operators as part of their inference engine: the "rules" of laws are no rules in the sense of production systems. The rules in an RM are the conceptualization (semantic structures) for expressing a regulation in natural language. The structure of rules is simple (see Den Haan (1996), and section 4). However, the resulting inter-rule structure may be rather complex, because it contains exceptions, which may be exceptions to exceptions etc. This structure is called the exception structure.

Figure 1 depicts the steps and their dependencies in this process. In the following sections (3 and 4) we discuss situation generation and rule generation, respectively. Finally, in section 5 two examples are presented.

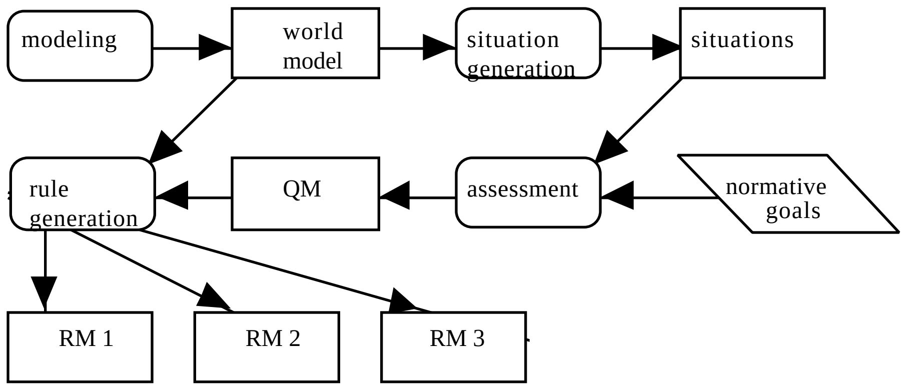

Figure 1: Generating regulation models from world knowledge and normative goals.

3 Generating situations

A qualification model (QM) is a further qualification of the set of possible situations that models a specific domain. The notion of enumerating the possible situations, and restricting that collection to meaningful situations is not entirely new in law Schadee (1957). He proposed to make a set of test cases for legislative drafts, that contain all the cases that are possible and meaningful according to the draft. Schadee recognizes the combinatorial explosion of possible cases by determining the number of terms in a domain, and estimating the number of terms per case. However, this analysis has not met any practical realizations or follow-up until now. By situation generation it is possible to generate such a set automatically, from the definitions of terms and by combining the possible states of objects.

First, we need an ontology that provides the definitions of terms van Heijst et al. (1996). In legal domains, human agents play a pivotal role, so we will distinguish two types of entities: (human) agents and objects. Further, agents and objects can be related in dynamic and in static ways. The dynamic relations are actions or processes, while the terms relation will denote all other types of relations.

Relations can involve various numbers of arguments, *i.e.*, agents and objects. A simplified definition of action that is sufficiently useful here is very similar to the notion of roles or cases in the semantics of verbs of natural language. This is a rather crude solution, but a more refined and better founded one derived from a “top ontology” will for our purposes not yield further relevant distinctions: see Hobbs (1995). This means that action terms have a number of slots which indicate roles: actor, object, recipient. The action implies (static) relations between agents. For instance, the action *give* implies that before the action takes place, the actor is in possession of the object. After the action, this relation holds between the recipient and the object. Moreover, there are constraints on the filling of the roles. Only agents can be the actor of actions; only forces can be the actor of processes. Actors may use forces, *i.e.*, a process may have an instrumental relation to an action. An attribute of the actor role is that an actor has the intention to have the action performed. It is assumed that the actor of the giving action has the intention that the object becomes in the possession of the recipient. Many other restrictions to role filling can be analyzed, but will not be discussed here. For normative reasoning intention and causality are not relevant anyway; they are relevant for attributing responsibility in law (Valente, 1995).

Therefore, in a simplified combinatorial view an action like *give* has two agents (actor and recipient) and one object (for the object role). Now, if we distinguish in the domain three types of agents, and four types of objects, 3 * 3 * 4 = 36 possible "giving"-situations are generated (This is not really correct. An action or process changes a situation into another situation. Therefore, one may assume that the give-action should be decomposed into an antecedent *situation1* in which an agent possesses an object, and a consequent *situation2* in which another agent possesses the object. Both situations are causally and intentionally related by the agent who is the actor of the change of possession (giving). This conceptualization is ontologically more correct, but does not really change the notion of combinatorics involved in situation generation. In this paper we treat actions as parts of situations; not as "bridges" between situations. The latter view complicates, but does not really change the processes of situation and rule generation. It is part of current research.). To illustrate what is involved, we may assume that situations consist of variables, agents (a), actions (c), objects (o) and relations (r). Furthermore, there are not only elementary situations that consist of one action, but also complex situations in which more actions or relations are involved (N_{cr} per situation):

• Number of agents: a
• Number of objects: o
• Number of actions: c
• Number of relations: r
• Average arity of action and relation predicates: n
• Average number of actions and relations per situation:N_{cr}

Each predicate may be an action or a relation (c + r). Each argument is is either an agent (a) or an object (o). Each predicate has arity n, so there are $(a + o)^n$ possible combinations of the arguments per predicate. This yields the total of possible actions and relations, which is the multiple (.) of (c + r) and (a + o):

$$Tot_{pred} = (c+r) . (a + o)^n$$

Suppose that a situation consists of an average of N_{cr} action or relation predicates, then the number of possible situations is:

$$Tot_{sit} = (Tot_{pred})^{Ncr} = ((c+r) . (a + o)^n)^{Ncr}$$

Filling the values of the traffic world into the equation above yields:

$$Tot = ((c+r) . (a + o)^n)^{Ncr} = ((20 + 20) . (15 + 20)^2)^2 = 2,401,000,000.$$

Indeed a classical combinatorial explosion. However, this is a worst case analysis, because we have only blindly applied the terms and types from the ontology and used an almost unconstrained version of the action frame (only the types of role fillers are given).

There are various sources for limiting the number of possible (and meaningful) situations (see also Den Haan, 1996).

- *Redundancy and tautology*
  There is a lot of redundancy when one does not carefully choose basic terms in representing knowledge Many relations may have tautological family members. It makes no sense to generate a relation *and* its inverse. For instance, in the traffic domain it is superfluous to see in-front-of(some-car, some-other-car) as different from at-the-back(some-other-car, some-car). However, as both relations are as good as one another, one should make some consistent decision as to which relations are the canonical ones. Another instrument for cutting down the number of situations is by exploiting symmetry. Crossings are symmetric, Therefore, one corner is sufficient to describe the spatial relations at all types of crossings.
- *Constraints*
  When the world knowledge contains descriptions of the physical limitations in the world, the physically and logically impossible situations are pruned. For instance, an object cannot be at different locations at the same time. Many of these are of a very general nature, like the roles in actions or physical principles, but there are often many domain specific ones as well. For instance, in the traffic domain, all actions are viewed as taking place in "only" two dimensional space.
- *Abstraction or grain size level*
  As in all modeling, and particularly in AI, abstraction is what keeps the world manageable, even if it leads to sometimes incorrect identifications at lower levels of grain size. If there are situations that enumerate all the subtypes of one super-type, then we can abstract all these situations into one generic situation that contains the super-type. If we are aware of some mismatch at a lower level, we may introduce *exceptions*. There is a trade-off in introducing exceptions and super-concepts, which is another way of stating that Occam's razor is applicable.
- *(Legal) Relevance*
  All modeling involves functional viewpoints along which abstractions are made. When describing a world some (configurations of) objects are explicit and stand out in a background of *assumed* present, but not relevant objects. In modeling worlds for legal (normative) control, worlds are only considered from the point of enforceable law. For instance, a traffic regulation is aimed at enhancing the safety of the participants.

Once the situations have been generated, they have to be distributed over two sets to form the qualification model QM. We mention three possible approaches:

1. Qualify each situation by hand according to the initial normative goals. An important source for qualification this way may be precedence law.
2. Formulate a limited number of situations of a high abstraction level, and qualify these by hand according to the initial normative goals or intentions.
3. Because legal drafting hardly ever occurs with a completely fresh start, one may qualify each situation using the older version of the regulation, using automated legal assessment. As in most political and technical debate the *differences* between the old and the new are emphasized, it is not difficult to use this method. followed by the first one.

For the relatively small domains we are using as our testbeds, qualification by hand is not yet a problem. Neither do we think that in practice one ever will have to rely only on this by hand method, but rather on the last one.

4 Generating rules

The input for the rule generation is the QM, which consists of the subset Σ^- of the undesired situations, and Σ^+ of the not undesired situations. $\Sigma^\pm$ is the normative default chosen. In law this is in general Σ^-, reflecting the principle that what is not forbidden is permitted in law. *s* is a situation. RM is the set of generated rules; *r* is a rule (for the representation formalisms for situations and rules, we refer the reader to Den Haan (1996)).

One may wonder why regulations are not stated as situations but as rules. There are two reasons. The first one is the same as for the world model: a long list of undesired situations is not a parsimonious, neither a coherent representation. On one hand, regulations model (un)-desired behaviour, but do not contain *every* possible situation because they should cover only a subset of the world model (Σ^- and some of Σ^+). On the other hand, regulations contain inter-rule structures of obligations/prohibitions vs. permissions (exceptions), which are the result of making abstractions that do not cover subsumed situations *completely* (note that the resulting exceptions are based upon the same abstractions (subsumptions) as the world model.). Abstract rules may cover more than one situation.

The second reason for translation concerns the translation of situations, which consists of conjunctions of states of objects and their relations, into a *more articulate format*. This is for pragmatic reasons. In linguistics, pragmatics mean conveying the intention of a text. Semantically, a legal statement expressed as a situation or as a rule is the same: they are paraphrases. A legal rule has two parts: a conjunction of conditions ("application ground") and a conjuncted deontically qualified action or state ("conclusion"). In a formal sense one may argue that the difference between a deontically qualified situation and a deontic rule is in the scope of the deontic operator: *e.g.*,

$$F(a \,\&\, b \,\&\, c \,\&\, d)$$
$$(a \,\&\, b \,\&\, c) \;\text{->}\; F(d)$$

We use here the conventional -> operator, although no real material implication is meant, but rather conjunction! There is only a commitment to state that F(d) is true when the application ground is true (see also Valente, 1995).

The overall algorithm for the generation of normative rules is defined as follows:

```
while $\Sigma \neq \{\}$
    select ($s$,$\Sigma$),
    translate ($s$,$r$),
    check_relation($r$,RM).
```

Here we do not further elaborate the **select** procedures as it is a simple set operation that selects a particular situation and removes it from a set. Below, we discuss the **translate** and **check_relation** procedures. As we will see, the application of these procedures may be nested: the adjustment may induce the need to formulate new or alternative rules, and they will also have to be added and call for adjustment. The construction of the rule generation procedures is based on machine learning techniques (see Kodratoff and Michalski, 1990).

The application ground is the "given" part of the legal statement, while the concluding part provides the *focus* of attention. This pragmatic articulation rhymes well with "if ... then ..." phrases. It provides the reader where s/he should pay attention. In a legal rule it often means the last, sufficient action that turns a legal situation (*e.g.*, (a & b & c & ¬d)) into an illegal situation. Both aspects of rule generation are covered by the **translate** algorithm proposed:

```
translate($s$, $r$)
    1) select_behaviour($s$, B)
    2) select_most_specific(B, b)
      3) if $s$ in $\Sigma^+$ then
      $r$ = ($s$ \b) -> O(b)
    4) if $s$ in $\Sigma^-$ then
      $r$ = ($s$ \b) -> F(b)
```

The first step in the translation algorithm proposed here is aimed at selecting the conclusion: some behaviour, B, which is in general some action ("ought-to-do" norm), but may also be some state ("ought to be norm"). The second step uses the subsumptions in the domain model in order to find specific subtypes of the behaviour: this may lead to "exceptions", and is thus instrumental in generating the exception structure of RM.

Following this, the relations between *r* and the other rules are examined:

```
check_relation($r$,RM)
while select($r'$ , QM)
  1) if not(imply($r$, $r'$ ))
     then
     2) if contradict($r$, $r'$ )
        then create_exception_rule($r$,$r'$ )
     3) if share_abstraction($r$,$r'$ ,level)
        then merge ($r$,$r'$ )
else add(r,RM).
```

In the first case, the situation for which *r* has been created already has been covered by another rule, because *r'* implies *r*. Therefore, *r* will not be added to RM. In the second case, the new rule *r* appears to contradict an existing rule in RM. The create_exception_rule procedure generates a permissive rule (P) when *r'* is a prohibition, or a restrictive rule (F), when *r'* is an obligation or a permission. If the conditions of *r* do not imply those of *r'* , but somewhere they share a mutual parent, *r* and *r'* are merged. The predefined abstraction level controls this procedure (see 3 below). When all the rules in RM have been inspected, but *r* does not relate to any of them, rule *r* is added to RM.

These algorithms are only the general frames without the complications and control that is required for putting the whole structure together. At least the following five parameters are involved in generating alternative versions of RM (see also Winkels & Den Haan (1995), Den Haan (1996))

1 *Normative default*

Selecting either Σ^+ or Σ^- to obtain largely obliging, or largely forbidding regulations. The former are in general the default in safety regulations where prescriptions are the rule, while in law the former is most often the case.

2 *Viewpoints*

The **select_behaviour**(*s*, B) procedure should be guided by consistent, but alternative sets of criteria, that reflect different viewpoints for drafting paraphrases. The pragmatics of rules allow one to put consistent focus on specific topics in a regulation. In particular, the conclusion parts of rules may contain selected behaviours that are consistent with some type of agent that should perform the actor role. For instance, a regulation can be drafted for a librarian and a paraphrase can be drafted for the borrowed.

3 *Abstraction level*

The current description of the adjustment algorithm creates regulations with the highest possible abstraction level, because rules are made as general as possible. To make less general rules, constraints can be introduced. For instance, one may choose to generalize only over one type of agent or action, instead of generalizing limitlessly (see rule 3 above).

4 *Deontic preference*

The deontic operators in the conclusions of rules may be predominantly O or F, and dependent on these: P. This paraphrasing consists simply of using the deontic rewrite rules.

5 *Preference for logical connectives*

Thus far we have only talked about conjunctions in rules. However, rules can be combined further by the additional use of disjunctions.

In a high abstraction level, one may decide to formalize a general rule as soon as two of the subconcepts are covered. In a medium-high abstraction level, one may choose to propose a general rule as soon as fifty percent of the subconcepts is covered. If a QM only contains so-called low-level situations, and no abstractions (see section 3), and one chooses the lowest possible abstraction level for the new regulation, then it can never be the case that a rule, which has been generated to cover one situation, "accidentally" covers another situation. The **adjust** procedure is only necessary when abstractions are involved (see also the second example in section 5).

5 Example

In this section we show how the procedures for the generation of situations and rules are executed by hand. First, we draw up the domain knowledge, and then situations are generated. We add the normative qualifications in order to generate the QM, and then we show the generation of two alternative RM. The example concerns owning and selling drugs. There are two types of agents in this example:

agent:	user
	pharmacist

and two types of drugs:

substance:	hash
	coke

Each substance may come in a particular quantity:

quantity:	5gr
	30gr
	100gr

There is one action:

action:	possess(actor:person,substance,quantity)

The *use* action concerns a substance and a person that has the role of actor - the drug addict in this case. For instance: *possess(user,coke,100gr*)

5.1 Situation generation

Each situation consists of one action. Since we know which parameters are used in the actions, we can now enumerate all the possible situations. This example does not model negations, because the world model is "closed": any behaviour that is not described, is not present in the situation. We abbreviate *cu* for user and *ph* for pharmacist.

possess(us,hash,5gr)	possess(ph,hash,5gr)
possess(us,hash,30gr)	possess(ph,hash,30gr)
possess(us,hash,100gr)	possess(ph,hash,100gr)
possess(us,coke,5gr)	possess(ph,coke,5gr)
possess(us,coke,30gr)	possess(ph,coke,30gr)
possess(us,coke,100gr)	possess(ph,coke,100gr)

The following situations are marked as undesired:

possess(us,hash,30gr)	possess(us,hash,100gr)
possess(us,coke,30gr)	possess(us,coke,100gr)

The rationale behind these qualifications is, that each user may have a small quantity of drugs for their own use. Pharmacists may own any amount of drugs.

5.2 Rule generation

In this paragraph we will apply the procedure for rule generation by hand. The first example is a low level regulation from the viewpoint of substances, and the second example is a high level regulation from the viewpoint of agents.

Low level regulation, viewpoint: substances

In the lowest abstraction level, we start with making rules for each situation in U (select and translate):

- Select situation, create rule
  r1= *F(possess(us,hash,30gr))*
- Check_relation: no rules yet
- Select situation, create rule
  r2= *F(possess(us,hash,100gr))*
- This process is repeated. We also find:
  r3= *F(possess(us,coke,30gr))*
  r4= *F(possess(us,coke,100gr))*
- Σ = {}, stop.

The resulting 7 rules can be abstracted over agents or quantities. We can perform abstractions as long as the normative contents do not change. Below, we have abstracted over quantity:

1 *It is forbidden that hash is in the possession of users in quantities over 5 grams.*
(merger of rules 1 and 2)
2 *It is forbidden that coke is in the possession of users in quantities over 5 grams.*
(merger of rules 3 and 4)

High-level regulation, viewpoint: agents

• Select situation:possess(us,hash,30gr)
• Check_relation: no rules yet
translate to highest abstraction:
    r1'=*F(possess(agent,drugs,quantity))*

Here, we see that if we choose a high abstraction level, we should also check whether new rules of a high abstraction level have unintended side-effects in terms of applying to situations from the wrong subset $\Sigma^{\pm}$ of Σ. This was not necessary in the previous example.

• Select situation from Σ^{+}: possess(us,hash,5gr)
• Check_relation: 2) create exception:
    r2 =*P(possess(us,hash,5gr)*
• Select situation from Σ^{+}: possess(us,coke,5gr)
• Check_relation: 2) create exception:
    r3=*P(possess(us,coke,5gr)*
• Check_relation: 3) r3 relates to rule r2, so merge them to
    r2' = *P(possess(us,substance,5gr))*
• This process is repeated for the pharmacist until we have the following rule:
        r3 = *P(possess(ph,drugs,quantity))*
• Σ = {}, stop.

The resulting rules are:

1 *It is forbidden to have any quantity of any drugs.*
1a *Users are permitted to have 5 grams of any drugs.*
1b *Pharmacists are permitted to have any quantity of any drugs.*

5.3 Analysis

When regulations model Σ^{-} on a high level of abstraction, the most abstract rules are prohibitions, while the exceptions are not prohibitions, *i.e.*, permissions. Regulations that are based on the lowest possible level of abstraction contain exactly those prohibitive rules that model Σ^{-}, while there are no exceptions. A regulation on a low level of abstraction implicitly states that all other behaviour is permitted: P(action(any, any, any)).

6 Conclusions

The procedures and algorithms described can handle relatively straightforward and simple domains: certainly an order of magnitude larger than the artificial one described in the previous section which consists only of 36 generic situations. The example also shows where further research is required.

• *Basic ontology for world models*
    The concepts we have used here to describe a world have not been really worked out. For instance, the possess action is a simple predicates here. Actions need to be added and far more structured so that it becomes obvious that actions presuppose antecedent states and have consequent states, so that it becomes *e.g.*, automatically obvious that selling implies buying and a new state of possession. By the way, analyzing actions and processes as changes of state, suggests that the distinction between "ought-to-

do" and "ought-to-be" norms is only pragmatic one, and not a semantic one. Similarly, the explicit representation of instrumental actions and processes should be further elaborated. Although further ontological commitments appear unavoidable, we will keep these as minimal as possible: not only to keep the scope of the representation formalisms as large as possible, but in particular to keep the modeling effort also minimal. The world model is only aimed at descriptive, not at predictive accuracy.

- *What's in a situation?*
  We have defined a situation as a simple list of states/actions. However, we may need to introduce some degree of articulation. For instance, instrumental processes should be part of actor roles, method/manner should be part of an action slot, etc. However, a more serious problem is the simple question of how big a situation should be. The description of a generic situation is restricted to only (legally) relevant states. However, relevance is a context dependent criterion (see Green (1989)). For instance, the fact that selling implies receiving money as a default is probably not relevant, but in the domain of trade it is. Relevance is also a factor in modeling in general and abstraction in particular. The more abstract the description, the less states one needs. For instance, because traffic is about moving on roads, the "moving on roads" situation is the most abstract one. The most abstract rule in traffic regulations is therefore that drivers should keep to the right (or to the left).
- *Viewpoints and parameters for rule generation*
  Rules provide an articulation on (subsets of) situations. The "if ... then ..." format is a typical "given-new" pragmatics construct (see Clark and Havilland (1977)). What is given (topic) and what is new (comment) is context dependent: not only dependent on the context of some discourse, but also in less dynamic contexts, such as particular groups of audience (citizens). In the example only the rule abstraction has been affected by viewpoint, but it may as well affect what is application ground and what is conclusion. For instance, if the substances are "given", but the sellers are "new", the rules should be of the kind: "if you sell coke, you have to be a pharmacist". This points to a second problem. Only some states or actions are under the control of (some of) the agents involved. One can only give a *specific* deontic qualification to those actions or states, and not to others. For instance, a situation that forbids that employees are at their work after 18.00 hrs could be translated as a rule that states that it is forbidden that it becomes 18.00 hrs if employees are at their work. Because actions are in general under the control of agents, selecting actions as conclusions is a good heuristic, but a very approximate one.

Current work is not only aimed at these issues, but also at the implementation of the algorithms. Special attention will be given to the creation of regulations of a high abstraction level. Section 5 showed that abstracting new rules instead of merging them (as is done in low level abstraction regulations) makes it necessary to look for exceptions outside the part of Σ that we are modeling (*e.g.*, looking into Σ^+ when we are modeling Σ^-). In this paper we have chosen to use the same procedures, but nested. This makes the generation algorithms less tractable.
The generation of paraphrases of normative regulations is not simply the automation of some subtask in drafting a regulation: it provides a functionality for which there is no equivalent by hand. Of course, one can try to do it by hand, but it easily becomes a full re-drafting exercise. More importantly, there is no guarantee that the hand-paraphrase is a real paraphrase, because human drafters may easily slip in unnoticed interpretative variances. However, one may wonder what problems this solution may solve. Is there a need for paraphrase of law? The need is in the first place in the various kinds of audiences for law. A traffic regulation aimed at children should not be highly abstract, but rather situation and role (pedestrian, bicyclist) centered. If citizens are supposed to know the law, the legislator is held to provide all reasonable and cost-effective means that this

knowledge is easily accessible (it is a shame that the Dutch government has not yet published its legislation (for free) at public networks: Internet) *and* understandable. Even if paraphrases are not geared towards specific audiences, some text structures are more easy to process than others. A second need is in the fact that any text - and legal texts are in this respect no exception at all - contains ambiguities. The QM is a semantically unambiguous version of a regulation and misunderstandings or interpretation debates can be avoided by reference to the QM, or by interpreting various paraphrases. Finally, the world model, the QM and the paraphrases may help drafters of regulations to obtain a more accurate, more complete and less ambiguously phrased regulation, for which the legal consequences are explicit and can be automatically tested.

References

Clark, H.H. and S.E. Havilland (1977). Comprehension *and the Given-new contract*. In: R. Freedle, editor, *Discourse Processes*, Ablex, Norwood NJ, 1977.

Green, G.M. (1989). *Pragmatics and Natural Language Understanding*. Lawrence Erlbaum Associates Publishers, Hillsdale NJ, 1989.

Haan, N. den (1996). *Automated Legal Reasoning*. PhD thesis, University of Amsterdam, 1996.

Haan, N. den, J.A. Breuker and R.G.F. Winkels (1996). Automated *legislative drafting*. In: *Proceedings of the NAIC 1996*, 1996.

Haan, N. den and R.G.F. Winkels (1994). The Deep Structure of Law. In: H. Prakken, A.J. Muntjewerff, and A. Soeteman, editors, *Legal Knowledge Based Systems: The Relation with Legal Theory* - Seventh International Conference on Legal Knowledge-based Systems Legal Knowledge-Based Systems, JURIX-1994, pp. 43-54. Koninklijke Vermande, 1994.

Hobbs, J.R. (1995). Sketch of an ontology underlying the way we talk about the world. *International Journal of Human Computer Interaction*, vol. 43, pp.819 - 830, 1995. Special issue on formal ontology.

Kodratoff, Y. and R.S. Michalski (1990*). Machine learning - An Artificial Intelligence Approach*, Volume III.. Morgan Kaufmann Publishers, San Mateo CA, 1990.

Svensson, J.S. P.J.M. Kordelaar, J.G.J. Wassink and G.J. van 't Eind (1992). ExpertisZe, a Tool for Determining the Effects of Social Security Information. In: C.A.F.M Grütters, J.A.P.J. Breuker, H.J. van den Herik, A.H.J. Schmidt, and C.N.J. de Vey Mestdagh, editors, *Proceedings of Legal Knowledge Based Systems*: Information Technology and Law, JURIX-1992, pp. 51-61, Lelystad, NL, December 1992. Koninklijke Vermande.

Schadee, H. (1957). Electronische wetgeving. *Nederlands Juristenblad*, vol. 3, pp. 53-58, Januari 1957. In: Dutch (English title: Electronic drafting of legislation).

Valente, A. and J.A. Breuker (1996). Towards principled core ontologies. In: *Proceedings of the Knowledge Acquisition Workshop 96*, Banff, CA, 1996.

Valente, A. (1995). *Legal Knowledge Engineering: a modeling approach*. IOS Press/Ohmsha, Amsterdam/Tokyo, 1995.

van Heijst, G., A.Th. Schreiber and B.J. Wielinga (1996). Using explicit ontologies in KBS development. *International Journal of Human Computer Interaction*, 1996. Special issue on ontologies in knowledge engineering, in press.

Voermans, W. and E. Verharen (1993). LEDA: a semi-intelligent legislative drafting-support system. In: *Intelligent Tools for Drafting and Computer-Supported Comparison of Law* - Sixth International Conference on Legal Knowledge-based Systems Legal Knowledge-Based Systems, JURIX-1993, pp. 81-94. Koninklijke Vermande, 1993.

Winkels,R.G.F. and N. den Haan (1995). Automated legislative drafting: Generating paraphrases of legislation. In: *ICAIL-95*, editor, Proceedings of the Fifth

International Conference on AI and Law, pp. 112-118, College Park, Maryland, 1995. ACM.