Automated Legislative Drafting: Generating paraphrases of legislation

Radboud Winkels and Nienke den Haan
Department of Computer Science & Law
Faculty of Law, University of Amsterdam
email: {winkels,nienke}@lri.jur.uva.nl

Abstract

In this paper, we describe which roles deep structures of law play in (automatic) drafting legislation. Deep structures contain a formal description of the intended normative effects of a new regulation. We discuss mechanisms that can be used to generate different paraphrases of regulations. Since it is possible to test the paraphrases on legal knowledge based systems, we have provided two extra design steps in legislative drafting which can be supported by automated tools.

Deep structures are straightforward descriptions of the normative effects of regulations. Each deep structure distinguishes desired and undesired behaviour, and has no further internal structure, such as paragraphs or exception structures. This paper describes methods to translate a deep structure into representations of different types of codes, i.e. paraphrases. Each representation of a code has a different surface structure, according to the choice we make during the translation regarding: a) the initial assumptions of the regulation, i.e. modelling from desired or undesired behaviour, b) the level of abstraction, c) the viewpoint of the law, i.e. the category of norm subjects, and d) the type of deontic modalities the regulation largely uses. All paraphrases have the same 'effects' as the deep structure, but with different features, and are suitable for different goals.

Keywords: deep structures, legislative drafting, legal knowledge based systems, codification

1 Introduction

In [denHaan & Winkels, 1994], we introduced the term *deep structure* of a regulation, to indicate the normative essence of the regulation that may be expressed differently at the surface after codification. Our use of the term 'deep structure' is inspired by, though different from, its use in linguistics by Chomsky, (e.g. [Chomsky, 1969]), to capture the semantics of language. It should also be distinguished from what some authors call 'deep models' or 'deep knowledge' in Legal Knowledge Based Systems (LKBS), e.g. [Bench-Capon, 1989; Breuker & denHaan, 1991], where deep knowledge systems have an underlying model that reflects the structure of the domain. Deep structures, as specified here, are a more formal and systematic description of ideas from legal theory (e.g. [vonWright, 1981; Rödig, 1976; Brouwer, 1990]), and AI & Law (e.g. [Hage, 1993; vanKralingen *et al.*, 1993]).[1] What we mean by a deep structure, is purely a description of the normative effects of a regulation. The contents of the deep structure lay below the rules of the regulation, hence the name 'deep structure'.

Deep structures of law are useful for:

1. Legislative drafting:
   - Allow different codifications with equal effects
     Section 5 shows how the deep structure is translated to several possible sets of rules.
   - Choose optimal codification
     The possible dimensions for generating different representations allow a motivated choice for a specific codification (see the examples of codifications in section 5).
2. Concentrate on normative effects of regulations
   As we will see in section 3, deep structures do not have the same complex structure as regulations. Deep structures show directly which behaviour is 'desirable' and which is not.
3. Prevent redundancies
   The descriptions of deep structures do not allow redundancies: a type of behaviour is either desirable, or it is not. When the directions of generation of paraphrases of section 5 are followed, it is possible – though not always desirable – to construct regulations without redundancies.
4. Law Comparison
   This aspect is not highlighted in this paper, but will be in future research. The bottom line is that comparing deep structures of regulations is more direct than comparing the regulations themselves (see points 2 and 3).

[1] One of the reviewers of [denHaan & Winkels, 1994] proposed the term 'virtual model', because the proposed model allows automated drafting and testing of legislation before the actual codification takes place. And indeed, we can see some correspondence to virtual modelling, for example in architectural design, where a design is modelled using computer graphics, and then inspected and tested. On the other hand, the term virtual model is much broader than we intend. Perhaps the term 'Basic structure' is an alternative.

Software/Knowledge Engineering:
goals & req. $\xrightarrow{1}$ conceptualization $\xrightarrow{2}$ formalization
$\xrightarrow{3}$ design $\xrightarrow{4}$ implementation
Legislative Drafting:
goals & req. $\xrightarrow{1}$ domain research $\xrightarrow{2}$ deep structure
$\xrightarrow{3}$ paraphrases $\xrightarrow{4}$ codifications

Figure 1: *Phases in software and knowledge engineering versus phases in legal drafting*

2 Legislative drafting

There are many requirements on the *structure* of regulations, and on the legislative design *process* (e.g. [Thornton, 1987; Waaldijk, 1985]). Each regulation should contain a description of its goal, contextual information regarding the relation to other regulations, and the date of enactment. In each country, there are several manuals on the *structural* part of legislative design. These manuals also contain the *correct procedure* to be followed in legislative design (cf. [Thornton, 1987]). In the Netherlands, [Waaldijk, 1985] and [Eijlander, 1993] are such manuals. In literature on legal drafting, we foremost see requirements on the layout of law texts, and there is no certified method for legal drafting [Eijlander, 1993]. There is a large gap between the description of the goals of new legislation, and the actual codification of new regulations. In this paper, we try to bridge this gap by using deep structures for law. In a deep structure, the requirements are translated to the desired normations on the possible behaviour a new regulation is meant to regulate. Next, the paper proposes transformation methods for translating deep structures into regulations, which can be tested automatically on legal knowledge base systems. These two processes, constructing deep structures and paraphrasing, should fill the gap in the legislative design process.

Just as formalization forms the bridge between 'conceptualization' and 'design' in both software and knowledge engineering, deep structures form the bridge between 'normative goals' and 'codification' (see Fig. 1). In system design, alternative designs are possible on the basis of the formalizations, which in turn can be implemented in a programming language (see for instance [vanVliet, 1984]). In legislative drafting, the deep structure can be translated into several alternative regulations. As the paraphrases of regulations are still formulated in a formal representation (see paragraph 5), we must subsequently codify them into natural language. The deep structure ensures that all alternative codifications regulate the same world in the same way, and we could employ each of them. The most 'computerized' approach we have found to legislative drafting is [Overhoff & Molenaar, 1991]. They use decision tables as a means for formalization, which is a very static way to represent legal rules and legal reasoning. We believe that the formalization of deep structures should be based on a clear seperation of situations in the 'world to regulate', and the normative qualifications of these situations.

3 Defining deep structures

The deep structure of new legislation defines how possible behaviour in the 'world to regulate' is qualified. In this respect, possible behaviour is only given the qualification 'desirable' or 'undesirable'.[2] The deep structure consists therefore of two parts: (1) a description of what is possible in a domain, and (2) a part that describes the legal qualifications. A deep structure is a description of all relevant possible behaviour, which is labeled either desirable or undesirable. *Structuring* these qualifications is a task which is performed during the construction of the representation of the new regulation, i.e. generating paraphrases.

In this paper we will use as an example the problems with the use of the computers of our department. We only want students of the Department of Computer Science and Law to use our computers, and we do not want them to use the machines for hacking, playing games or commercial use.

1. Model of the world to regulate
   - Terminological knowledge
     Objects, agents and actions, with attributes. The isa hierarchy for our small example is given in Fig. 2.[3]
   - Behavioural knowledge
     Causal/intentional relations between objects and actions, e.g. in our domain:
     $take\_course(Cs, P) \leftarrow Course(Cs) \wedge Person(P)$
     $use(C, P, U) \leftarrow Computer(C) \wedge Person(P) \wedge use(U)$
   - Structural knowledge
     Structural relations between objects or actions (e.g. consist-of and spatial relations). In our domain we could specify: *keyboard* is part of *computer*.
2. Normative qualifications of situations

   The normative qualifications as expressed in legal rules, generally attach norms to 'generic' situations. A situation is a collection of behaviour descriptions in the world. An example in the world of computer use, would be the use of the terms: { *use(Person, Computer, Use), take_course(Person, Course)* } in a primary rule. Whereas we see specific descriptions of situations (instances) in cases, legal rules contain more general descriptions because they have to apply to classes of objects, e.g. computers, as opposed to instances. An example of a specific situation in a case might be: { *use('Emma', 'SPARC_station_ELC', playing), take_course('Emma', 'Dutch')* }.[4]

   The world knowledge describes all (relevant) possible behaviour in the world. We define a generic situation σ in the world as a non-empty set of formula's about the world:

   $\sigma = \{F_1, F_2 ..., F_n\}$ where $\sigma \neq \emptyset$

   where a formula F has one or more variables, of which at least one is free (ungrounded). The set of all possible generic situations in the world is called W_p.

   For our example domain we can specify these as:

[2] At present it is unclear whether it is necessary or desirable to represent other categories like 'rights' and 'duties' in deep structures, cf. [Hohfeld, 1919]. This is one of the things we need to examine in the future.

[3] We are not concerned with attributes in this example.

[4] In applying the legislation to specific situations, the specific situation in a case is compared with the generic desscriptions in primary rules. For this paper we are not concerned with actually applying legislation, but see [Valente & Breuker, 1994] for a formalization of that process.

$\forall_c \forall_p \forall_{cs} \forall_u \exists!_{W_p} \forall_\sigma (\sigma \in W_p \leftrightarrow$
$(Computer(c) \wedge Person(p) \wedge Course(cs) \wedge Use(u) \wedge$
$\sigma = \{use(c,p,u), take\_course(cs,p)\}$

which reads as: W_p is the (only) set of all generic situations σ, which in turn are sets of all behaviours *use* and *take–course* which are defined over all computers, persons, courses, and uses respectively.

The deep structure of legislation is an abstraction of the legal rules, and therefore contains generic descriptions and normative requirements. For each normative system there has to be a default normative requirement for a situation in the world. The default can be either 'legal' or 'illegal' (or 'desirable' versus 'undesirable').[5] This distinction is expressed as: "Everything is permitted but..." versus "Everything is forbidden but...". In the first case the regulation describes the world of undesirable situations (W_u), in the second case the world of desirable situations (W_d).

Modelling W_u: prohibitions

In most cases the default will be 'legal', i.e. if in a particular regulation nothing is said about a particular situation, it is a legal situation. E.g. in our example:
1. It is forbidden to use the computers of the CSL dept. for non students
2. Students may not use the CSL computer for playing games.

Modelling W_d: permissions/obligations

In some cases the default will be 'illegal'. For instance in safety related domains the risks at stake may be so high that in principle all behaviour is undesirable except for some limited prescribed courses of action. In such a case the regulation prescribes those actions (using obligations) and all others can be considered illegal. E.g. [Hammond *et al.*, 1994] describe the domain of safety protocols for clinical tests of medicine like chemotherapy treatment of cancer. The risk of killing a patient by wrong treatment should be minimized as much as possible. In our domain an example might be:
1. All students may use the CSL computers
2. All use of the CSL computers should be educational

For the remainder of this paper, we will aim at a regulation that only allows students of our department to use our computers for educational purposes. We can formalize this in two different ways, starting with W_d or starting with W_u. Reasoning from the desirable state, we allow situations where our own students use our computers for educational means. We do not pose any restrictions on the use of other computers than our own. W_d only consists of two generic situations:

$$
\begin{aligned}
W_d = \{ & \\
& \{CSL\_Course(Cs), Student(St), \\
& CSL\_Computer(C), take\_course(Cs, St), \\
& educational\_use(C, St)\} \\
& , \\
& \{Non\_CSL\_Computer(C), Person(P)\ , \\
& Course(Cs), use(C, P)\ , \\
& take\_course(Cs, P)\} \\
& \}
\end{aligned}
$$

If we look at the restrictions on the computer use of students, we see that we want to forbid students to use our equipment for Non_educational purposes. Reasoning from the undesirable state, we therefore describe situations in which our students play, hack, or use our computers for commercial means. Secondly, we forbid anyone else to use our computers. W_u consists of four generic situations:

$$
\begin{aligned}
W_u = \{ & \\
& \{CSL\_Course(Cs), Student(St), \\
& CSL\_Computer(C), take\_course(Cs, St), \\
& playing(C, St)\} \\
& , \\
& \{CSL\_Course(Cs), Student(St), \\
& CSL\_Computer(C), take\_course(Co, St), \\
& hacking(C, St)\} \\
& , \\
& \{CSL\_Course(Cs), Student(St), \\
& CSL\_Computer(C), take\_course(Cs, St), \\
& commercial\_use(C, St)\} \\
& , \\
& \{CSL\_Computer(C), Person(P), \\
& Course(Cs), take\_course(Cs, P), \\
& use(C, P)\} \\
& \}
\end{aligned}
$$

The intersection of W_d and W_u is empty, and so it should be.

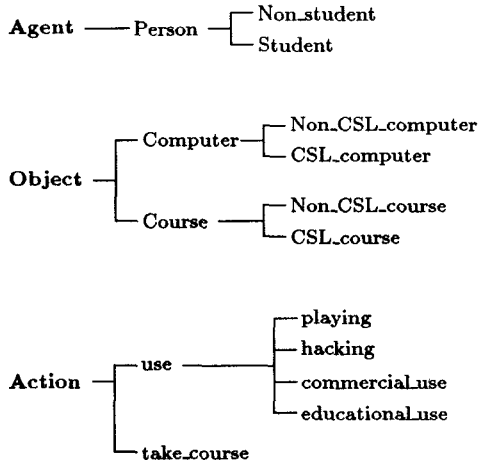

Figure 2: *The isa hierarchy*

4 Mapping a deep structure to a paraphrase

The regulations we are to construct, have to reflect the normative descriptions as defined in the deep structure. We have the following building blocks at our disposal:

- Permissions and obligations stipulate desired behaviour.
- Prohibitions qualify undesired behaviour.
- The terms from the world description are used in defining the rules.
- We can form exception structures by using combinations like obligation/prohibition, or prohibition/ permission. Law texts often contain a general rule which prescribes a norm for a large group of individuals, actions or objects, followed by a rule that excludes specific members or subsets of these groups. The exception can be made *directly* in the same article, or

[5] One can also decide to have the default 'silent', i.e. when a regulation does not say anything about a certain situation.

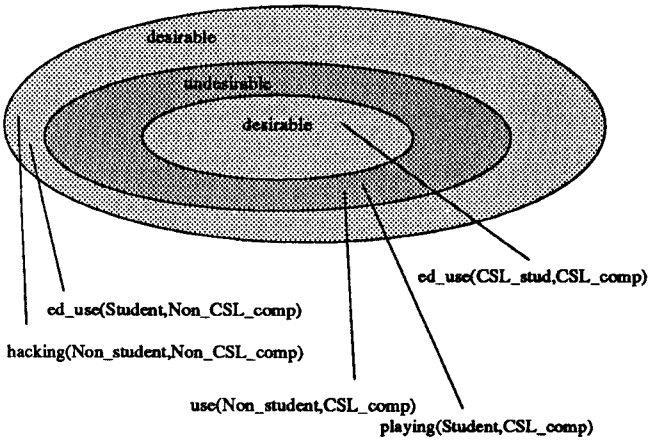

Figure 3: *Venn diagram; light grey areas are W_d.*

indirectly in another paragraph or even another regulation. Furthermore, the exception can be made *explicitly*, characterized by words like 'contrary to' and 'except', or *implicitly* where the exception has to be inferred. For example, in our domain the requirement: "Only students may use our computer equipment, but not for commercial means" could be codified as: "art 1.a: Only students may use the computers of the CSL department", and "art 1.b: Use of the CSL computers for commercial means is prohibited". As an indirect and implicit exception, we may state in another paragraph on system management: "art 23: The system manager may grant permission to an individual to use the CSL computers for a specific goal". Article 23 acts as an implicit exception to article 1.a (and even 1.b), because any individual, and not just students may be granted access to the CSL computers (apparently for any use). This exception has to be inferred at the domain level from the agent hierarchies of computer users. Conflicts between rules because of such exceptions are solved by 'secondary rules' [Hart, 1961]. For instance, in our domain we could define a domain specific secondary rule that states that decisions by the system manager prevail over general rules.

- We may use references between rules if definitions are repeated.

In Fig. 3, we see a simple Venn–diagram of some situations in our example world. In our regulation, we want to allow or oblige the elements in the light grey areas, and forbid the elements in the dark grey areas. If we model W_u, then the first layer of norms are prohibitions (or deontic equivalents). Only these modalities can restrict the default general permissibility. Permissions in the first layer are superfluous. Exceptions to prohibitions, are permissions or obligations (or their deontic equivalents). Exceptions to obligations or permissions, are prohibitions or their deontic equivalences ($Fx = O\neg x = \neg Px$).

In representing desired behaviour (W_d), we can use obligations, or permissions. Both denote desirability, but obligation is much stronger than permission. To be able to choose between the two modalities, we have to study the set of desirable behaviour. If there are positive examples of the particular behaviour, but also negative, then we can model the desired behaviour with a permission. When there are only positive examples, then the behaviour must be obliged. For instance, when we see an area of W_d where students are using the computer for educational purposes, and also for non-educational purposes, then educational use is clearly permitted but not obliged. Exceptions to obligations or permissions are again represented as prohibitions (or their deontic equivalents). Given a deep structure of the intended legislation, we can start to generate different paraphrases reflecting this deep structure.

5 Generating paraphrases

We can construct different structures in regulations, if we model along one, or combinations of the following dimensions:

1. Default undesirable/desirable: *restrictions/ prescriptions*
   The first choice that has to be made is whether the *default* will be 'illegal' or 'legal', whether we will describe W_d or W_u. The normative part of the deep structure is already based on a choice, but in principle it is possible to generate a set of rules that starts from the opposite assumption. For some purposes it may even be advisable to describe both W_u and W_d, e.g. if the legislation should be understandable for small children that need to know both what is permitted and what is forbidden, even though one could infer one from the other. In most cases, at least some rules will refer to situations that are actually covered by the default for ease of understanding.

2. Viewpoint (according to knowledge types in world): *legislator/norm subjects/...*
   We have to choose a *viewpoint* for describing normative positions of situations. The different possible viewpoints correspond to the different types of knowledge in the world model of the deep structure. The viewpoint can be based on the agents and objects in the world, on the possible behaviour, and on structures in the world. For our example domain, we could describe the deep structure from the perspective of using computers (i.e. what use is desirable, what use is not), from the perspective of users (i.e. which users can use the computers, which can't), or from the perspective of computer use. From the latter perspective we can either describe the prohibitions on playing, hacking and commercial use, or forbid to do anything else but educational use. The results of different choices are logically equivalent, but can be dramatically different on the surface. One viewpoint may for instance result in mostly explicit exceptions, another in mostly implicit exceptions. This again has consequences for the readability and learnability of the legislation. Explicit exception structures are easier to read and learn for humans than implicit ones.

3. Abstraction level
   high: few rules generate all normative qualifications
   low: a rule for each legal qualification
   A third dimension for generating different expressions of the same deep structure is the level of *abstraction* one chooses for the surface structure. When we have chosen a particular level of detail, we can only make additions on lower levels which can be abstracted to the chosen level. E.g. if we have two instances a and b which can be abstracted to c, and the normative

effects Fx and Fy which can be abstracted to Fz, we can make the following abstractions:

$\{a \rightarrow Fx, b \rightarrow Fx\} \overset{abs.}{\rightarrow} a \vee b \rightarrow Fx$ or:

$\{a \rightarrow Fx, b \rightarrow Fx\} \overset{abs.}{\rightarrow} c \rightarrow Fx$

We can also concatenate different normative effects for the same set of situations:

$\{a \rightarrow Fx, a \rightarrow Fy\} \overset{abs.}{\rightarrow} a \rightarrow Fx \vee Fy$ or:

$\{a \rightarrow Fx, a \rightarrow Fy\} \overset{abs.}{\rightarrow} a \rightarrow Fz$

On the one extreme, one can simply enumerate all desirable or undesirable generic situations[6], on the other extreme one can try to describe them at the highest abstraction level possible. For some purposes, concrete rules that specifically refer to all kinds of situations and define normative positions for them, are more suitable, for other purposes abstract rules may be preferred.

4. Deontic transformations: *O-based/F-based/P-based*
A fourth mechanism for generating paraphrases is using *deontic transformations*, e.g. $P(x) = \neg O(\neg x)$ and $F(x) = O(\neg x)$. Besides literally using the negation in some situation ($\neg x$), the domain (world) taxonomies can be used for explicitly stating the negation of (x), e.g. in our example domain:
$O(educational\_use(c, st))$ can be restated as:
$F(hacking(c, st) \vee commercial\_use(c, st) \vee playing(c, st))$
besides $F(\neg educational\_use(c, st))$.
This last choice yields yet another paraphrase based on the fact that the world is assumed to be closed and the number of subtypes of '*use*' in this case is finite.

Using these, and possibly other mechanisms it is theoretically possible to generate numerous – and if we take different natural language formulations into account, an infinite number of – paraphrases of a deep structure of a normative system. Below we present two examples of codifications generated for the deep structure of our example domain, based on the mechanisms described above.

Example 1

1: default: undesirable
2: viewpoint: computers
3: abstraction level: high
4: deontic: permission

1.0 All use of all computers by anyone is forbidden. (DEFAULT)
1.1 The use of CSL computers for educational purposes by students that take a CSL course is permitted.
1.2 All use of non-CSL computers by anyone is permitted.

Example 2

1: default: desirable
2: viewpoint: students
3: abstraction level: low
4: deontic: permission

2.0 Anyone can use all computers for all purposes. (DEFAULT)
2.1 Students that take a CSL course may use the CSL computers.
2.2.1 Students may not use the CSL computers for playing.
2.2.2 Students may not use the CSL computers for hacking.
2.2.3 Students may not use the CSL computers for commercial use.
2.3 Students that do not take a CSL course may not use CSL computers at all.
2.4 Non-students may not use the CSL computers at all either.

The two sets of rules concerning the use of the computers of our department, look very different at first sight. Closer examination reveals that, given certain restrictions on the world of computers, their users and possible use, the two sets may boil down to the same thing.

The formalization we can use for primary rules provides us with applicability grounds (conditions) and normative descriptions (conclusions). The conditions of the primary rules correlate to the prescriptions of the sets in W_d, whereas the conclusions of the primary rules must yield only the desired normations from W_d. Since we take the world of desirable behaviour as our starting point, the behaviour mentioned in the definition of W_d is permitted behaviour. The formula's in the generic situations provide us with the applicability grounds of the possible rules. The most abstract rule we can formalize stays close to the deep structure:

$\forall_c \forall_{st} \forall_{cs} (CSL\_Computer(C) \wedge Student(St) \wedge CSL\_Course(Cs) \wedge take\_course(Cs, St) \rightarrow$
$Permitted(use(C, St, educational))$
$\forall_c \forall_p \forall_u Non\_CSL\_Computer(c) \wedge Person(p) \wedge use(u) \rightarrow$
$Permitted(use(c, p, u))$

All permissions must be exceptions to a general rule. We took W_d as a starting point, so all the other situations are forbidden (W_u). Quantifying over the computers is useful when the regulation is drafted for the staff of the department. For the students, quantification over Student can be selected. The level of abstraction over facts is determined by choosing a level in the hierarchical knowledge. We can say that educational_use is permitted, but equally, we can enumerate that playing, hacking and commercial_use are forbidden:

$\forall_p \forall_c \forall_u Person(p) \wedge Computer(c) \wedge use(u)$
$\rightarrow Permitted(use(c, p, u))$
$\forall_p \forall_c \forall_u Person(p) \wedge CSL\_Computer(c) \wedge use(u)$
$\rightarrow Forbidden(use(p, c, u))$
$\forall_{st} \forall_c \forall_{cs} \forall_u Student(st) \wedge CSL\_Computer(c) \wedge use(u)$
$\wedge take\_course(cs, st) \wedge CSL\_Course(cs) \rightarrow$
$Permitted(use(c, st, u))$
$\forall_{st} \forall_c \forall_{cs} Student(st) \wedge CSL\_Computer(c) \wedge take\_course(cs, st) \wedge$
$CSL\_Course(cs) \rightarrow$
$Forbidden(use(c, st, playing))$
$\forall_{st} \forall_c \forall_{cs} Student(st) \wedge CSL\_Computer(c) \wedge take\_course(cs, st) \wedge$
$CSL\_Course(cs) \rightarrow$
$Forbidden(use(c, st, hacking))$
$\forall_{st} \forall_c \forall_{cs} Student(st) \wedge CSL\_Computer(c) \wedge take\_course(cs, st) \wedge$
$CSL\_Course(cs) \rightarrow$
$Forbidden(use\_use(c, st, commercial))$

The first rule starts high in the taxonomy of persons and use. The second rule and third rule form the permitted state (W_d), in which the exceptions are written out in the last three rules. For the abstraction of the last three rules to educational use, the terminological knowledge concerning these actions must naturally be present explicitly. This results in the following rule:

$\forall_{st} \forall_c \forall_{cs} Student(st) \wedge CSL\_Computer(c) \wedge CSL\_Course(cs) \wedge$
$take\_course(cs, st) \rightarrow$
$Forbidden(\neg use(c, st, educational))$

[6] Of course, in reality this is not feasible for most domains, since they are too large and complex.

6 Conclusions and future research

We described a 'deep structure' that lies below the surface codifications of regulations, that captures the normative essence of the regulations. In a way, the process we describe of first defining a deep structure, and next generating paraphrases, is a reversal of the *possible worlds semantics* approach to deontic operators. In that approach, possible and ideal worlds are used to interpret deontic notions like obligations and permissions. We define the ideal world (directly as W_d, or indirectly through W_u) first, and use it to generate formulas or rules that use deontic operators to describe it.

The main benefit of our approach for the legal world, is that deep structures provide a safe testing ground for new legislation. Artificial intermediate models are also found in virtual reality, for instance in industrial design, to test the strength of architectural constructions, or to simulate process control in chemical plants. In this context, we also see that there are different options that support the same predefined qualifications, for instance, there are several ways of supporting a roof that fit the same architectural taste. Different codifications are possible for the same legal qualifications by using combinations of the four dimensions. This not only increases the usability, but also the explainability of regulations. We expect that the fact that we will be able to generate paraphrases automatically, and that we can test them automatically, will be a substantive contribution to drafting legislation. We have some experience in (automatic) testing of regulations. In the TRACS project, a fraction of the Dutch Traffic regulation has been tested on some traffic situations. In [denHaan, 1993], the process of testing is described as:

1. Generation of test cases
   Terms form the world description are combined to form a description of a situation. Any combination of agents and actions is possible. We can either pick elements by hand, as was done in the prototype system TRACS, or design an automated generation of test cases that enumerates all possible combinations of elements.

2. Automated application
   The regulation has been represented in a legal knowledge based system. The regulation is applied to all the test cases.

3. Validation of the legal qualifications
   In the TRACS prototype, validation took place by hand. When a deep structure of the regulation has been defined that corresponds to the external requirements, then we can compare the test results and the deep structure automatically.

4. Refinement of rules
   In [denHaan, 1993] refinements are proposed to alleviate any mismatches between the qualifications and the desired effects. The paper proposed a system for abstraction and specification of rules that would transform the applicability or normative effects to fit the desired effects of the regulation. In the context of deep structures we may use this mechanism to optimize the deep structures, but *not* the representations. Any change in the deep structure can automatically be translated to a representation.

The deep structures for law have also practical consequences for law comparison. Looking directly at the effects of regulations is very helpful for comparing two versions of the same law, or similar laws from different countries. The regular rule structures of laws may be very dissimilar, which makes the comparison all the more difficult. The deep structures provide an interlingua for the comparison. Law comparison is more effective when we compare the deep structures instead of the full regulations, because they directly describe the normative effects and are independent of the internal design of regulations. The deep structures help us to overcome differences in abstraction levels, viewpoints, or other modelling conventions such as the choice of deontic modalities or the desirable/undesirable default. One of the aims of law comparison may be harmonization of legislation. For instance, the European Community aims at standardizing its legal systems. We can combine the deep structures of different national regulations, i.e. form the union of the sets of desirable/undesirable behaviour. The chance of generating different codifications with the same legal effects may increase the suitability of this new legislation for different countries.

At this stage, we are working on a description of the process of 'deep structuring' and on specifying software tools for generating paraphrases. The Dutch Association for Scientific Research (NWO) has just granted a research fund for the development of this methodology and tools.

References

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

J.A. Breuker and N. den Haan. Separating world and regulation knowledge: where is the logic? In M. Sergot, editor, *Proceedings of the third international conference on AI and Law*, pages 41–51, New York, NJ, 1991. ACM.

P.W. Brouwer. *Coherence in Law: An Analytical Study.* PhD thesis, University of Leiden, Groningen, The Netherlands, June 1990. In dutch.

Noam Chomsky. *Deep structure, surface structure, and semantic interpretation.* Bloomington, Ind., 1969.

N. den Haan and R.G.F. Winkels. 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*, pages 43–54. Koninklijke Vermande, 1994.

N. den Haan. Towards support tools for drafting legislation. 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.* Koninklijke Vermande, 1993.

Ph. Eijlander. *De wet stellen. Beschouwingen over onderwerpen van wetgeving.* W.E.J. Tjeenk Willink, Schoordijk Instituut - Centrum voor Wetgevingsvraagstukken, 1993.

J. Hage. An information network for legislative engineering. 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*, pages 43–52. Koninklijke Vermande, 1993.

P. Hammond, J. Wyatt, and A. Harris. Drafting protocols, certifying clinical trial designs and monitoring compliance. In J.A. Breuker, editor, *Proceedings of the workshop Artificial Normative Reasoning at the European Conference on Artificial Intelligence*, pages 124–131, Amsterdam, 1994.

H.L.A. Hart. *The Concept of Law.* Clarendon Press, Oxford, 1961.

W.N. Hohfeld. *Fundamental legal conceptions as applied in legal reasoning.* Yale University Press, 1919. Fourth printing, 1966.

Mr. R.W. Overhoff and Mr. L.J. Molenaar. *In de regel beslist. Een beschouwing over regelgeving met behulp van beslissingstabellen.* SDU Uitgeverij, Plantijnstraat, 's-Gravenhage, 1991.

Jürgen Rödig. Logische Untersuchungen zur Makrostruktur rechtlicher Kodifikate. In Jürgen Rödig, editor, *Studien zu einer Theorie der Gesetzgebung,* pages 592–611. Springer-Verlag, Berlin, Heidelberg, New York, 1976.

G.C Thornton. *Legislative Drafting.* Butterworths, London, third edition edition, 1987.

A. Valente and J. Breuker. A commonsense theory of normative systems. In J.A. Breuker, editor, *Proceedings of the workshop Artificial Normative Reasoning at the European Conference on Artificial Intelligence,* pages 56–68, Amsterdam, 1994.

R. van Kralingen, E. Oskamp, and E. Reurings. Norm frames in the representation of laws. 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,* pages 11–22. Koninklijke Vermande, 1993.

J.C. van Vliet. *Software Engineering.* Stenfert Kroese B.V., Leiden/Antwerpen, 1984.

G.H. von Wright. On the logic of norms and actions. In Risto Hilpinen, editor, *New Studies in Deontic Logic,* pages 3–35. D. Reidel, Dordrecht, 1981.

Mr. C. Waaldijk. *Wetgevingswijzer.* Koninklijke Vermande B.V., Lelystad, 1985.

Acknowledgements The ideas about deep structures in law have materialized in discussions with Joost Breuker of the Department of Computer Science and Law. We would also like to thank Henry Prakken for his valuable comments on an earlier version of this paper.