Isomorphic Modelling of Statutory Law

Versus Top-Down Modelling

Jørgen Karpf *


PREFACE

The paper concerns modelling of statutory law. The point of view is long-term, and is of greatest interest if the statutory area of law is large. The paper describes and compares two different methods of representation of the complex interaction of statute sections.

Isomorphic modelling starts out with the statutes and has the objective to represent each section separately in a clause. The method is neutral and universal as regards applications. Top-down modelling, on the other hand, starts out with a number of queries or applications of the law. The queries are broken down to calls of notions in the statutes. The notions are represented by combining contents from different sections.

Isomorphic modelling requires a representation language with higher order facilities for a clause to alter the evaluation of another clause and for priority of clauses. Semi-isomorphic modelling can be carried out with say Prolog.

The methods are compared by a number of quality measures for modelling of

* M. Sc., LL.M. - Department of Computer Science, Technical University of Denmark, DK-2800 Lyngby, Denmark.
law. In addition, some general recommendations are given, in particular modularization.

1. INTRODUCTION TO MODELLING OF LAW

The paper presents some of the results of a project on legal knowledge-based systems at Copenhagen Business School and Thechnical University of Denmark. See Karpl, Legal Logic Programming Techniques, 1988.

1.1. Jurisprudential background

It must be noted, that Danish jurisprudence is based on Scandinavian legal realism, cf. Preben Stuer Lauridsen, in Gammeltoft-Hansen et al, Danish Law, 1982. An account for the legal sources in the Danish legal system and their priorities is beyond this paper. Briefly stated, however, there is a general agreement that there are four sources of law, namely legislation, precedent, custom and the nature of the case. A general theory of the binding effect of precedents is not recognized. However, Danish courts treat relevant precedents with great respect.

For the particular jurisprudential background of modelling of law, applied in this paper, confer especially: Eckhoff, Rettskildelære, 1985; Sundby, Om normer, 1974; Eckhoff & Sundby, Rettsystemer, 1976.


In the Danish legal system, and especially in precisely demarcated areas of law suitable for development of knowledge-based systems, legislation, including secondary regulations, is the dominant legal source. For this reason, from a Danish point of view, modelling of statutory law is considered the principal task in modelling of law.

1.2. A principle of human-computer cooperation

It shall be noted that the ultimate assessment of correctness involves an evaluation of the whole situation when a designated user applies a knowledge-based system in law. Certain aspects of reasoning in law, such as most common sense reasoning, evaluation of fairness, discovery of particular and not foreseen issues of a case, will be reserved for the human user. This is expressed in the following principle of cooperation.

Principle of division of labour between a human user and a law computer system – The labour shall be divided between the human and the computer system such that the strengths and weaknesses of the two parties complement each other.
1.3. Stepwise modelling of law

Modelling of law may be carried out in several steps, leading from the original legal sources, through complete intermediate models in natural or formal languages, and finally arriving to a knowledge-based computer system. This general and recommendable method is called: stepwise modelling. An earlier term was progressive modelling. The main objective of stepwise modelling is to preserve the fine-grained structure of the reasoning in a law domain. The method may be followed, regardless of the choice between isomorphic or top-down modelling. (Top-down system development is also called stepwise refinement. This term must not be confused with stepwise modelling).

1.4. The four main tasks in modelling of law

The following tasks are considered to be the main tasks in modelling an area of law that is suitable for development of a knowledge-based system:

1. Representation of statutes and secondary regulation
2. Representation of precedents
3. Representation of open-texture concepts and discretionary issues
4. Assurance of coherence of the total model, possibly assisted by representation of principles in law.

Isomorphic modelling addresses in particular the first main job consisting of representation of the statute sections and their interaction. It does, however, also provide a frame for two of the other jobs: representation of open texture concepts and discretionary issues and representation of precedents.

Assurance of coherence of the total model, including the user cooperation with the model, is in part addressed in paragraph 8 on quality measures.

Representation of legal principles, such as *lex specialis legi generali derogat*, stating that a specific law abrogates more general laws, is an advanced research area. Cf. Hamfelt & Barklund, 1989.

2. Modelling of statutory law

2.1. Isomorphic and Top-down modelling of statutory law

In the project on legal knowledge based systems at the Copenhagen Business School, parts of the Danish Inheritance Act has been modelled by means of

---

Prolog. A main idea has been to investigate how far it is possible to model a statute with Prolog, so that:

The statute is represented in a separate module, consisting of separate representations of each section in the statute.

In the paper a section means:

— an ordinary statute section,
— a subsection of a statute section, or
— a part of a subsection, e.g. a period or a sentence.

In an isomorphic model of a statutory area of law, each statute section is represented separately in a strict sense. Its representation is called a clause. The term «clause» from logic and logic programming is used in a broader sense then usually. However, in some of the arguments below, it is assumed that a clause has the structure of a rule with a conditional part and a conclusion.

The separate representation of each statute section implies, for instance, that if a section contains a main rule and is tacit about an exception to this rule contained in another section, then the clause for the first section may not have a reference to the clause for the other section. In the model, the exception shall be accomplished solely by the clause for the other section and by the inference mechanism of the representation language.

This language will thus need higher-order facilities, so that the evaluation of a clause may be altered afterwards or beforehand through the evaluation of another clause. The reasoning process here has resemblance with non-monotonic reasoning.

A semi-isomorphic model also has separate representations of statute sections. In the representation of a section, its tacit use of another section may, however, be accomplished by the insertion of a call to the clause for the other section. Semi-isomorphic modelling is possible in say Prolog.

By isomorphic modelling, a statute module will have the same structure as the text of the statute. This and subsequent statements about «isomorphic modelling» also holds, unless otherwise stated, for «semi-isomorphic modelling», but possibly with minor modifications.

The isomorphic method is neutral as regards applications and allow for any «conceivable» concrete query.

Top-down modelling of statutory law starts out with a number of standard queries or applications of the law. The queries are broken down to calls of notions in the statutes. These notions may be represented by combinations of contents from different sections. A main rule and its exceptions may, for instance, be represented in a single clause.

The methods are compared by a number of quality measures for modelling
of statutory law. The choice between the methods is most important when
the modelled statutory area of law is large, i.e. when it has a large number
of sections. Confer the large scale applications reported in Bench-Capon et

A new project, in collaboration with the Technical University of Denmark,
on development of a representation language for isomorphic modelling is
briefly introduced in paragraph 9. The conclusion of the paper is that such a
development must be an important aim in the research on modelling of
statutory law.

2.2. Modularization

Modularization is an integral part of isomorphic modelling. For each statute,
the statute module shall consist solely of clauses corresponding to sections of
the statute.

Queries are evicted to a query module. In isomorphic models, in principle
any query containing calls to several sections from a number of statutes can
be posed. It should not be allowed for a query directly to represent a notion
from a statute. Instead the query may call the representation of the notion,
as defined by one or several section clauses. A query module can have a
number of ready for use standard queries.

For a typical model developed by the top-down method, the query and the
statute modules are integrated.

Management of facts is evicted to the facts module. The statute module can
generally assume that necessary facts are available.

General system controle consists of: access controle, various execution
controles, such as controle of logging and level of explanation, file
management, document management, security routines, statistics, etc. These
controle functions may not be part of the modules above, but are collected
in a controle module, or possibly several controle modules. Customization
for different user categories may be assembled in exchangeable user modules.

Modularization may provide the system with a general controle of flow. The
user enters a query through the user module to the query module. This
module calls the statute module to have section clauses evaluated. When a
fact is needed, the facts module is called. This module will either find the
fact in a database, or through the user module ask the user to supply the
fact. At this stage, the user may be assisted with decision support or
information on relevant precedents. Normally, when all the calls in the
query have been evaluated, the user will receive the answer to his query
from the query module through the user module.

Acquisition of facts is accompanied by facts validation. A comprehensive
validation is performed by the facts module. Examples: a person can have at
most two parents, and no person is his own descendant. Extra guidance and control may be performed in a user module. An example: a person has normally two parents, and only if the user is positive, will the system accept a situation, where a person legally has zero or one parent.

2.3. **Representation of open texture concepts and discretionary issues**

There are several groups of open texture issues, and their classification is not the subject of this article. In connection with statutes, however, there are probably three main classes:

1. Is a fact in a concrete case inside or outside the domain of the fact description in a section of the statute?
2. Do some of the facts of the case qualify for a certain legal standard used in a section of the statute?
3. Which consequence shall be derived in a concrete case, when a section of the statute has a range of possible conclusions?

A fourth class could deal with questions such as if the case qualifies for special treatment by some general legal principles. The classification does not pretend to be complete. It has one purpose: to show that for an area of law governed by statutory law, an open texture concept or a discretionary issue, belonging to one of the classes above, very often corresponds to a description in a statute section.

For these open texture issues, the statute module in cooperation with the facts module will serve as a skeleton upon which special treatment can be coupled in. The simplest treatment is to have the user answer the questions with only rudimentary support.

An advanced system may have several decision support modules, each having its own features and qualities. One module may support classification of hierarchical structured facts, another module may use fuzzy logic and a third one Bayesian logic. Each module has two parts: a general part and a part consisting of specific information for each open texture issue the module is designed to give support for. A number of systems have been developed with hyper-text support facilities.

For many issues the best decision support is via search for precedents.

2.4. **Representation of precedents**

Precedents in a statutory area of law can roughly be grouped in two categories:

1. Precedents with decisions with high relevance for interpretation of a single statute section.
2. Any other precedent of relevance for interpretation of the statute.
The first group of precedents are particularly suited to be included in a information system coupled unto the skeleton of the statute model.

The second group may be subdivided in several ways, for instance by relevancy for interpretation of chapters of the statute.

An important open texture issue for which there are many precedents can be given decision support by means of an inductive system, e.g. an algorithmic inductive system or a neural network. This is especially an opportunity in many administrative applications, where administrative precedents are included in the basis for the decision. Cf. Karpf, Inductive Modelling in Law, 1991.

3. Interaction of Statute Sections

Each section of a statute is a formulation of usually a fragment of a norm. In this context, the distinction between directives and norms of competence is not of concern. Norms of qualification, however, are of special interest. Notions about legally qualified facts are dealt with in the treatment of norms of qualification. Notions about general facts are represented as calls to the facts module. In the subsequent, a notion means a legal notion. In a model of predicate logic, a notion is represented by a predicate with a number of variables.

It is mandatory to arrive at distinct definitions of the following concepts:

A section employs directly a notion from outside the section.
A section is directly affected by a notion from outside the section.

A section is said to directly employ a notion from outside the section, if it has an explicit and direct reference to this notion. The notion may be defined in one or several other sections. If in one of these sections, the (partial) definition of the notion directly employs another notion, then the former section at most employs the latter notion indirectly (unless, of course, the former section does have an explicit and direct reference to the latter notion as well).

A section is said to be directly affected by a notion from outside the section, when this notion is defined in one or several other sections, and if either it is explicitly stated in one of these sections, that the notion shall affect the first section, or the direct affect is otherwise evident or in accordance with legal practice.

Examples of this kind of interaction are: high priority rules, norms of qualification and rules of exceptions. It occurs that a section contains an explicit and direct reference to a notion, which even without this reference directly would affect the section. The two concepts can be combined into:

A section directly depends on a notion from outside the section, if it either directly employs the notion, or directly is affected by the notion.
Two methods of describing the section interaction shall briefly be mentioned.

3.1. The graph for interaction of statute sections

The first method is to define a directed graph for the statute section interaction. Two sections S and T are connected with an arc from S to T, if S directly depends on a notion defined in T. There are two types of arcs: one type expresses that S directly employs a notion from T, and the other type expresses that S is directly affected by a notion from T. An initial section in a statute is a section with no other section depending on it. Typically, an initial section deals with a fairly high-level legal issue. On the other hand, a final section is independent on notions defined in the rest of the statute. The premises in a final section contain calls of factual notions.

The initial sections in the graph will always correspond to sections that formulate notions ready to be used in queries (or in other statutes). In principle however, queries may be based on any notion in the statute. As stated in paragraph 3.2, isomorphic representation makes it feasible to represent a query based on notions from any sections in the statute.

A further analysis of the graph involves treatment of (indirect) dependency and analysis of transitivity of the dependency relation, the possibility of cycles, etc.

Analysis of transitivity of dependency – If S directly depends on T, and T directly depends on U, then it does not necessarily follow that S is really dependent of a notion from U. The dependency may have disappeared in the passing of the intermediate section T. This is because sections are still fairly large units of meaning. The result is that dependency, affection or employment are not transitive relations.

Possibility of cycles in the graph – Usually – at least for a small and uncomplicated statute – it will be the case that the graph for a statute is acyclic. However, that may not always be the case. A counterexample may be found in a recursive definition, e.g. in a legislation concerning limited companies, where the relation between a parent company and daughter or subsidiary companies may be formulated in such a way that the graph involving these sections will have a small cycle. Besides such intended, recursive instances, a cycle in a graph can not be taken as an indication that the statute contains an inconsistency, cf. the remark above about the non-transitivity of dependency. For a complex area of law regulated by a number of statutes, there is a theoretical possibility of an involuntary cycle involving inconsistency.

96
3.2. Reformulation into a punctilious statutory text

The second method is to reformulate a statute into a punctilious statutory text. This is suitable as a preparation for isomorphic modelling. It is carried out by section and has the following steps:

1. Identify the instances when the section directly depends on an outside notion. Identify the section(s) in which the notion is defined.
2. Check if either the section has an explicit reference to the notion, or if a section defining the notion has an explicit statement that the notion shall affect the former section.
3. If necessary, elucidate the existing text or add an explicit reference in the former section or an explicit statement in the latter. The requirements for a reference or statement to be explicit may or may not include numbers of the involved sections.
4. Check if the reference to the notion, or statement about its affect, is accompanied by a sufficient description of the usage of the notion. If necessary, elucidate the existing text or add an explicit description of the usage of the notion.

4. The idea of isomorphic modelling

The idea of isomorphic modelling of a statute is very straightforward and must have occurred to everyone working with modelling a number of statute sections.

4.1. Previous work and isomorphic modelling

Sergot et al, 1986, Bench-Capon et al, 1987, Biagioli et al, 1987 and Sherman, 1987 report on projects where semi-isomorphic modelling may have been used, but it is difficult to judge to which extent.

However, until recently there were no extensive reports of the idea as a modelling method, involving a separation between query clauses and section clauses, so that a query clause is composed of calls of section clauses, and so that it is prohibited for a section clause, except by calls, to include any content from other sections. Since a preceding version of this paper, a couple of articles have been published addressing the issue of structure-preserving modelling methods. Cf. for instance Bench-Capon 1989, Routen 1989 and Bench-Capon & Coenen 1991.

This may be because of the impediments for the method, especially high-priority sections and sections that alter other sections. Faced with these impediments, I suppose developers generally may have resorted to weakly enforced semi-isomorphic modelling, various ad hoc methods to overcome the impediments and/or shifted their modelling method towards...
top-down development based on a selected number of queries. In much of the reported modellng, the focus has been on some of the many other important aspects of modelling, than that of finding an optimal method for representing interaction of statute sections. Many modellng tools – maybe most shells – have not the facilities to make a try for isomorphic modellng. To fully realize isomorphic modellng, a special purpose, maybe higher order representation language is needed.

4.2. The term «isomorphic modellng»

In the start, I called the method «programming of a statute article by article». See Karf, Legal Logic Programming Techniques, 1988. When it was discovered, that Prolog is inadequate for a true realization of the method, and that the method has implications beyond the mere programming, the name was changed to «representation of a statute by section» and later to «isomorphic representation of a statute». It is now realized that the method has important implications for almost all parts of modellng an area of (statutory) law, including representation of open texture concepts, discretionary issues and precedents, hence the name: «Isomorphic modellng of law».

«Isomorphic» means «of equal form». The word «homomorphic» has the broader meaning «of alike form». In algebra and other structured fields of mathematics, a homomorphic function preserves structure, and it is isomorphic, if it is also injective, in the sense that the function applied to two different objects shall give two different values. Modellng by any method has a similar objective: two cases that in the real world has different solutions, ought also by their representations in the model to get representations of different solutions. For this reason, the name «isomorphic» is preferred to the alternative «homomorphic».

5. ISOMORPHIC AND SEMI-ISOMORPHIC MODELS

Isomorphic models may be defined by a description of the isomorphic modellng method, constituting an internal description of an isomorphic model.

Alternatively, isomorphic modellng may be defined by the functioning of the resulting model, as perceived by the collection of different user categories. This constitutes an external functional description of an isomorphic model. The external description may be derived by the quality measures for modellng, cf. paragraph 7.8. The two ways of description supplement each other.

The internal description:

A model of a statutory area of law is called isomorphic if:
1. The representation of a statute – the statute module – is the result of the following:

a) The text of the statute is disassembled into the smallest contiguous units feasible for representation, each unit containing a single norm or a fragment of a norm. The disassembly of the text into these units shall be a subdivision of the existing division of the statute into chapters, articles, sections and subsections. For the sake of convenience, the units are called sections.

b) Each section is represented separately by a clause.

c) The statute module consists of the assembly of these clauses.

2. The separate representation of a section by a clause shall be based upon a natural linguistic interpretation of the section.

When a section directly employs a notion outside the section, its clause shall contain a call of the representation of the notion.

When a section is tacit about a direct affect from a notion outside the section, its clause may not contain a call of the representation of the notion. Instead, the clause for one of the sections defining the notion must have an altering affect on the clause for the former section. It is the last two sentences that constitutes the difference between a full realization of isomorphic modelling and semi-isomorphic modelling.

The above may by no means be a hindrance for interpretation of the sections in accordance with the relevant national legal doctrine, including other legal sources, all in order to obtain correct functioning of the statute module.

3. The statute module is separate from the representations and management of: queries, facts, decision support, precedents, controle functions and user customozation. It is a further demand that these functions when appropriate are coupled unto the skeleton of the statute module. By this demand:

The structure of the statute is propagated to other appropriate parts of the model.

6. IMPEDIMENTS TO ISOMORPHIC REPRESENTATION

There are two general types of sections that directly affect evaluations in other sections, and as such may be called impediments to isomorphic representation. The two types are:

1. A section which may be described to have high priority in relation a number of sections, maybe even all other sections in the statute.

2. A section which otherwise alters the evaluation of one or several other sections.

A high-priority section does alter the evaluation of a number of other sections. The alteration of the other sections are, however, achieved in a
uniform way. For any of the sections to be altered, the alteration may, for
instance, be described to consist of addition of an extra foremost condition
in the conditional part of the rule fragment of the section.

A section altering only one or a few other sections may prescribe a much
more complex alteration. Actually, an alteration may be as complex as
natural language permits, only governed by the will and skill of the drafters
of the statute.

In the subsequent, treatment of a section as a high-priority section implies
search of a representation of the section with a uniform description of the
alterations of the sections in its scope.

On the other hand, treatment of a section as one which otherwise alters the
evaluation of other sections may involve individual descriptions of each of
the alterations.

Treatment of a section that directly affect evaluations of other sections as a
high priority section or as belonging to the second type is a matter of
expediency.

6.1. High priority sections

When a section is said to have higher priority than another section, the
latter will tacitly make use of a notion, often a condition, from the former.
The following examples are described according to a typical priority hierarchy
of a statutory law.

Interjurisdictional law – A first group of high-priority regulations of the use of
the statute or part of the statute contain international or interprovincial
limitations. Often this regulation is not part of the statute. The information
may be supplemented in another extra section as part of a punctilious
reformulation of the statute, if this is considered convenient. The decision
of whether or not say Danish law applies for a certain matter in a case, is
often a complex issue. Typically, the decision involves issues about the
Danish territory, Danish nationality and/or a number of other points of
association with Danish law. The choice of jurisdiction may be another
candidate for decision support.

Temporal regulation – A second group of high-priority regulation are temporal
regulations, also called effect regulations, consisting of rules regulating the
time when a statute or part of it shall come into effect or cease to be in
effect. For some statutes, the regulation of when the statute shall come into
effect is not part of the statute itself, but may be a part of the enactment
procedure. Ceasing to be in effect for the whole or part of a statute is often
regulated as part of the enactment of a new statute or in an amendment act.

In a reformulation of the statute into a punctilious statutory text, it may be
convenient to add a special extra high-priority section containing any effect
regulation concerning the statute, which is not already contained in the existing sections.

*Amendment acts – Amendment acts* to the statute (or a number of statutes) consists of usually several instances of the following kinds:

- addition of a new section to the statute;
- altering of a section;
- deletion of a section.

These changes may all take effect at a certain time, or the changes may take effect at different times. A *temporally complete* edition of a statute up until say January 1, 1991, will contain the statute in its original form supplemented with amendment acts for the statute and the necessary temporal regulations from the first enactment of the statute and until the said date.

In contrast, more ordinary use of time in single section of a statute may for instance be of the form, that a provision only shall apply for a matter, for which a certain event has taken place after some given date. This ordinary use of time comparisons are quite similar to other conditions in the conditional part of a rule, and should not be treated in a special way.

Only temporal conditions that are part of the effect regulation for the statute or parts of it, ought to be included under the term temporal regulation.

The accumulated temporal regulation may if convenient be collected in a special high-priority section. It is often complicated to decide upon which regulation shall apply:

- the law prior to the first enactment of the statute,
- the statute as originally enacted, or
- the statute as amended by a number of amendment acts.

For many areas of statutory law, deciding upon this issue may be a candidate for decision support. A case for which the answer is ambiguous, concerning the question as to which provisions apply, is a candidate for either exclusion from real treatment by the system, or for hypothetical treatments.

The formulation of a section of a statute almost invariably tacitly assumes that the section is in force. The objective is that its isomorphic representation shall be equally tacit about this issue.

*Rules on qualification etc.* – A third group of regulations has lower priority than interjurisdictional and temporal regulations, but higher priority than the bulk of the statute. It contains *general rules concerning the application of the statute*. It may, for instance, be *norms of qualification*.

An example is the Danish 1972 Adoption Act. Section 16 of the act
contains a statement that adoption from the date of October 1, 1972 or later has the consequence that the child, in all legal respects, completely changes family by the adoption: The child obtains the same status as a biological child of the adopters and the legal tie to its natural family is severed. (The Danish Inheritance Act section 4 also has an explicit reference to the regulations of adoptions). This norm of qualification must, with its temporal limitation, be used in all the terms involving family relations in the Inheritance Act. The sections in the Inheritance Act are formulated by means of terms for ordinary family relations, so it is indeed tacit in these sections, that these terms shall be interpreted to apply the modified relations.

It is very common for a section of a statute to contain a *legal definition, clarification or specification* for a notion applied elsewhere in the statute. The scope for such a section may be the whole statute, a chapter of the statute or some enumerated sections. The affected sections are usually formulated in such a way that it is only tacitly that such legal definitions etc. shall be applied.

If the scope of the section containing say a legal clarification is large, it will, if possible, be convenient to treat the section as a highpriority section. If the clarification implies alterations that are difficult to describe uniformly, the section may instead be treated as a section that otherwise alters the contents of other sections.

6.2. Sections otherwise altering evaluations in other sections

When drafting a statute or an amendment to a statute, the three main reasons for using the technique of having a section alter the content of one or several other sections probably are:

1. To avoid *repetitions* of, say, a qualification of a notion. To separate the qualification of a notion from its applications also gives an increase in the *clarity* of the text.
2. To reduce the complexity of a section by evicting treatment of a *variety of exceptions* to other sections.
3. To conveniently describe minor amendments to existing sections.

A section containing a definition or a qualification of a notion should, if possible, be treated as a high-priority section. However, this demands that the representation language has facilities to describe the implied alterations in the affected sections in a uniform way. The use of the notion in the affected sections may be so variant that one must resort to individual descriptions of the alterations, a procedure in weak violation with the idea of isomorphic representation.

It is therefore an objective to achieve such an expressive power of the representation language that a qualification of a notion can be represented
by one high-priority clause, regardless of the different usages of the qualified notion elsewhere in the statute.

**Rules of exceptions** – Exceptions come in all kinds. The three major ones are:

— an exception concerning the conditional part of a rule;
— an exception concerning the consequence of a rule, and
— an exception concerning both the conditional part and the consequence of a rule.

A rule of exceptions may itself have exceptions.

Prolog has the following very limited priority facilities:

— the order of evaluation of clauses for the same predicate;
— the order of evaluation of premises in the conditional part of a clause, and,
— controle of ordering by usage of the cut-operator.

In Prolog programming, these facilities are often used to handle exceptions. The method of handling exceptions by means of high priority clauses is worth an examination.

However, the natural evaluation process for a reader of the statute text is probably first to evaluate an issue by the main rule followed secondly by evaluations of the issue by the rules of exceptions with a possibility of having the first result altered. Ordinarily, this is also the section order of the rules.

If the representation language inverts the natural order of the evaluations, the transparency may suffer severely. When the acquisition of facts is query-driven, a query involving a main rule with a number of exceptions represented as higher priority clauses, will result in an awkward sequence of facts acquisition. The system will first ask for facts that may qualify for the exceptional handling, and only at last will it ask for the rest of the facts needed in the main rule. The explanations during this process will likewise be awkward.

These considerations point towards a representation capable of preserving the natural order first to evaluate the main rule and then to evaluate the associated rules of exception.

6.3. **Distinction between preaffects, intraaffects and postaffects**

The following is an alternative or a supplement to the distinction between high priority sections and sections that otherwise alter evaluations in other sections.

When a section is directly affected by an outside notion, in most cases, the affect can be classified as:
1. A *preaffect* with a beforehand change of the evaluation of the section,
2. A *postaffect* with an afterwards change of the evaluation of the section, or
3. An *intraaffect* by which the change of the evaluation is more intricate
than in 1 or 2.

A preaffect can often be conceived as an additional precondition foremost
in the conditional part of the clause for the altered section.

A postaffect containing extra limitations can equally be conceived as an
additional postcondition in the end of the conditional part of the clause for
the altered section.

A postaffect that shall enlarge the solution set of the clause to be altered
may possibly be conceived as a disjunctive premise in the end of the
conditional part of the clause.

It should be possible to make specifications for an isomorphic representation
language that can handle preaffects and postaffects. However, especially
concerning postaffects, there are problems concerning usage of existentially
qualified variables in the clause to be altered.

Intraaffects may be much more intricate to handle.

7. **Top-down modelling**

This modelling method could also be called *query-driven modelling*.

It is characterized by an effort to model the area of statutory law with
special consideration for a number of applications, very often with regard to
a certain user category.

In contrast, customizing an isomorphic model to a certain user group,
leaves the statute module unchanged, and consists of suitable changes in the
query module and development of a user module for the group.

Top-down modelling can more detailed be described by:

1. Identification of the queries the model shall be able to answer.
2. Development of the model in a top-down fashion, as well and efficient as
possible with respect to these queries. In this process, at most secondary
considerations are given to allow for other queries than the selected ones, or
to preserve the structure of the statute.

If it is deemed convenient, notions from several sections are combined and
represented in a single clause. High priority of a section may simply be
achieved by calls to its clause foremost in the relevant queries. This is an
example, in which part of the meaning of the statute is implemented in the
selected queries. It is therefore not the case that the clauses for the statute
carry the meaning of the statute not even the represented part of its
meaning. The clauses for the statute will not exhibit the same form as the statute text.

Top down modelling has important advantages. The two most obvious ones are that:

— the method can be employed today with a great number of development tools, and
— even in the near future, the method will, probably, be much faster than some circumstantial isomorphic modelling, leaving more time for other parts of the modelling.

In the last comparison, however, the gain in programming speed may be lost by more time-consuming correctness verification and documentation.

The following paragraph contains a number of quality measures of models and modelling methods. Besides the two advantages of topdown modelling cited above, isomorphic modelling is assessed to be superior by a number of measures.

8. Quality measures of modelling

A modelling method must foremost be measured by the qualities of the models developed by the method, but also the celerity and costs of the method are important. In a research project the latter aspects are at first of minor importance. If a method produces models of significantly higher quality, tools may be developed to increase its speed and reduce its costs. Only measures of particular relevance for legal modelling are mentioned below. It must be admitted that the selection of criteria, and certainly the comments on the criteria, are biased towards the isomorphic modelling. A model or a modelling method may be measured by the criteria:

1. Correctness verifiability
2. Applicability
3. Transparency and user acceptance
4. Amendability, extensibility and compatibility
5. Various criteria and generality of the method

As stated in paragraph 1.2., the ultimate assessment of correctness involves an evaluation of the whole situation when a designated user applies a knowledge-based system in law. In the sequence, a more narrow concept of correctness of a model is investigated.

8.1. Correctness verifiability

For an isomorphic model, correctness verification of the functioning of statute module can be described as having two parts. The first part is to ensure that each section is represented in a correct way. This can,
theoretically and practically, be tested one section at a time. The other part consists of an overall examination to ensure completeness and consistency of the statute module. During this examination at the latest, incompleteness and more seldom inconsistency in the statute itself must be captured and dealt with. However, the proper interaction between the clauses is supposedly taken care of by the facilities of the representation language.

The verification has two characteristics:

(i) Its first part is very well defined, and its second part is probably as well defined as generally possible.
(ii) The extent of the verification work is probably reduced to a minimum, especially when the universal applicability of the model is taken into account.

For a model developed by any other method, the correctness verification is less well defined. For a model developed by the top-down method, a procedure of testing and correctness verification in accordance with this method will be used. It has, however, no absolute meaning to verify the correctness of an individual clause, be it a query clause or another clause representing part of the statute. These clauses are constructed arbitrarily by the developer, often as a representation of a combination of parts from different sections. It is also possible that incompleteness in the statute will escape capture. Therefore:

(i) The correctness verification is less well defined, than in the situation above.
(ii) The extent of the verification work is probably larger than above, due to the arbitrary changes of the structure in the model, in relation to the structure of statute.

Lawyers may have a share in the practical part of the testing of a model. For an isomorphic model, a technically minded lawyer, but not necessarily with any programming experience, may even participate in the internal, theoretical correctness verification of the statute module by reading the source code of the statute module and directly compare it with the sections of the statute. It is an aim of the specification of an isomorphic representation language, and for the programming by means of the language, to achieve this high degree of legibility:

Lawyers shall be able to read and understand a statute module and point out possibly incorrect clauses.

A model developed in say Prolog probably fails in this measurement.

(As a remark in passing, the functioning of the total model shall, regardless of its kind, be tested by a number of methods. One of these is to develop a corpus of test cases that systematically tries out if possible all the different paths the execution may follow).
8.2. Applicability

A model has high applicability if it allows for a great number of applications or queries. If it allows for any conceivable concrete query, it may be called neutral and universal as regards applications. For abstract queries, see 7. A model may have a restricted selection of queries – for instance if it is developed with a compiled Prolog. If addition of a new query is a matter of adding a new clause to the query module, and then recompile, the model still has high applicability. For a model with high or universal applicability, it may be deemed necessary that the user module for a certain user category only allows for and gives support for a selection of queries. High or universal applicability implies that the model by exchange of user module may be used by widely different user categories.

8.3. Transparency and user acceptance

Transparency is a measure of the extent of revealment of the interior working of a model, or equivalently, the extent of user understanding of the behaviour of the model. A detailed comparison between isomorphic models and other models with respect to transparency is beyond this article. Two aspects are treated: documentation and explanations.

Documentation of the total model is facilitated by modularization. For an isomorphic model, documentation of the source code of the statute module may consist of a two-column listing: the statute text in one column and the corresponding clauses in the other column. Another possibility is to have the statute sections as comments directly in the source code. Additional comments can be needed. (This is indeed the case if an incompleteness in the statute has necessitated the programmer to complete its representation by creation of a clause, that is not a direct representation of a notion in a statute section). In comparison with any other modelling method, the isomorphic method can indubitably have the highest possible quality of documentation. As stated above, it is within range for lawyers to actually read even the source code of the documentation.

During the use of a model, an explanation generator can answer how- and why-questions. If at any time, the actual execution is part of an evaluation of a section clause, explanations can be generated using phrases identical or similar to the text in the section. It is comparatively simple to develop the following explanation facility, consisting of an optional mode of execution. In this mode, the execution is stepwise, and a law window shall at any time show a segment of the statute text, containing in high-lighted letters the section corresponding to the section clause presently under evaluation.

Conceptual model and traditional legal reasoning – Only one additional aspect of user acceptance will be mentioned. A user always creates his own conceptual model of a computer system, and this model differs more or less
from the system. If the user already has a conceptual model for the application area of the system, he will use this model as a basis for his formation of a conceptual model for the system. This is good, if the system works in accordance with the prior concepts of the user, and if not, it may delay or even block the user from getting a satisfactory conceptual model of the system.

For a lawyer and anyone else familiar with the statute, an isomorphic model constitutes the best basis for formation of a conforming conceptual model.

8.4. Amendability, extensibility and compatibility

Amendability is a measure of the ease of change of a model in connection with an amendment act. This represents a leap in nonmonotonicity. It can be incorporated in two ways, either by extension of the model with a representation of the amendment act and of the temporal regulations, or by modification of the clauses for the amended sections combined with addition of clauses for new sections. Extensibility is a measure of the ease of enlarging the model to cover a new (part of a) statute. Compatibility measures the cooperation ability of several models, even to the extent of forming a conjunction of models.

8.5. Various criteria

Suitability for university usage: Two user categories are special: law students and law scholars. They both need versatile models. They deserve the whole range of benefits only offered by isomorphic models. They will, however, need very different user modules. For these categories, representation of abstract cases serves a purpose. This is a major challenge, and it should probably not be undertaken, but in connection with the punctilious isomorphic modelling.

Authentication of legal models may become an option, or even compulsory for some models in say public administration. Isomorphic models are superior in this respect.

Finally about the ability to have decision support and integrated information system for precedents, isomorphic modelling demands that the structure of the statutes is propagated to the whole model. Literature and the precedents of relevance for the modelled area of law are in a number of respects intimately connected with and indexed by the structure of the statutes. Therefore, decision support for open texture issues as well as an information system for precedents can be of higher quality in isomorphic models than in other models.

Generality of the method – Isomorphic, semi-isomorphic and top-down
modelling are all general methods. The two former are more general being neutral as regards applications, whereas the latter depends on a selection of queries for some applications. If this selection at some stage proves insufficient, much work may have to be redone. Also during the representation of a statute, top-down modelling inflicts arbitrary divergencies in the model in comparison with the structure of the statute. In contrast, isomorphic modelling may be called canonical.

The external functional description of an isomorphic model – An external, functional description of isomorphic modelling supplementing the internal description can be derived from the criteria in 8.1 - 8.5. Whenever a design issue shall be decided upon, the decision may affect one or several of the criteria, e.g. the transparency of the model. An example: If exceptions to a main rule are represented as high priority clauses, this will invert the natural order of evaluation and result in an awkward order of facts acquisition and explanations. The example illustrates that the functional description effectively supplements the internal description.

Isomorphic modelling shall imply encouragement of procedures that improve the stated quality measures.

9. REPRESENTATION LANGUAGES FOR ISOMORPHIC MODELLING

In the autumn of 1988, a new project was started in collaboration with professor Jørgen Fischer Nilsson, Department of Computer Science, Technical University of Denmark. Nilsson has recently specified and developed a prototype for a higher order logic language. It has been named LEIBNIZ. It is distinguished by absence of variables. Language expressions take form of ground terms forming a combinator logic. Cf. Nilsson, 1988 and 1989.

The aim, briefly stated, for our project is:

1. To further categorize the forms of section interaction that are impediments to isomorphic representation of a statute in 1. order predicate logic.
2. To specify extensions to 1. order predicate logic, so that the impediments can be overcome by representation in the extended logic.
3. To develop a prototype isomorphic logic programming system based on the extended logic, probably by means of a tailored version of LEIBNIZ.
4. To develop, using the isomorphic logic programming system, a couple of small scale, truly isomorphic models of suitable areas of statutory law.

10. CONCLUSIONS

1. Isomorphic modelling is superior to top-down modelling by a large number of quality measures.
2. Isomorphic modelling is only realizable by a special purpose representation language with higher order facilities.
3. Development of an isomorphic representation language must be a major aim in the research on modelling of statutory law.

BIBLIOGRAPHY


J. Hansen, Simulation and automation of legal decisions, Complex, Oslo, 1986.


ICAIL 1991. Proceedings of The Third International Conference on Artificial


J.F. Nilsson, Knowledge Base Property Combinator Logic. ISSN 0902-2821, Department of Computer Science, Technical University of Denmark, 1988.


A. Ross, Om ret og retsfærdighed. En indførelse i den analytiske retsfilosofi, 2, oplag, København, 1966.


N.K. Sundby, Om normer, Universitetsforlaget, Oslo, 1974.

