Artificial Intelligence and Law **5:** 207–242, 1997.

A Method for Conceptualising Legal Domains

An Example from the Dutch Unemployment Benefits Act

PEPIJN VISSER[1], TREVOR BENCH-CAPON[1] and JAAP VAN DEN HERIK[2]
[1]*Department of Computer Science, University of Liverpool, P.O. Box 147, Liverpool L69 7ZF, United Kingdom*
E-mail: {pepijn, tbc}@csc.liv.ac.uk
[2]*Department of Law and Computer Science, University of Leiden, P.O. Box 9521, 2300 RA Leiden, The Netherlands*
E-mail: jfricd@ruljur.leidenuniv.nl

Abstract. There has been much talk of the need to build intermediate models of the expertise required preparatory to constructing a knowledge-based system in the legal domain. Such models offer advantages for verification, validation, maintenance and reuse. As yet, however, few such models have been reported at a useful level of detail. In this paper we describe a method for conceptualising legal domains as well as its application to a substantial fragment of the Dutch Unemployment Benefits Act (DUBA).

We first discuss the intermediate models (called expertise models), then present a three-stage method for their construction, drawing on the COMMONKADS work in knowledge acquisition, conceptual models of statute law, and the KANT method of knowledge analysis. Subsequently, we describe how these techniques were applied to the DUBA, and provide detailed examples of the resulting model. Finally, conclusions on the framework and guidelines are given as well as means of recording and presenting the various design choices.

Key words: conceptual models, system design, ontologies

1. Introduction

As experience with constructing legal knowledge-based systems (LKBS) has grown, it has become increasingly evident that their disciplined construction requires that we first build a model of the knowledge to be incorporated at a level higher than that of executable code. Originally there were hopes that the declarative nature of the languages used to build LKBS would enable a transition from source knowledge, whether in the form of texts or elicited from an expert, to executable code with no intervening steps. But these hopes proved false; transformation into executable code requires a number of decisions to be made: selection, interpretation, choice of entities and attributes, level of abstraction and the like, and it is important that these decisions be recorded explicitly, rather than needing to be reconstructed from the code. The need for proper design and documentation is present for any properly-engineered system, whether an LKBS or a conventional system. It is now usual to attempt to produce intermediate models of the systems (cf. the idea of building

a knowledge-level model (Newell, 1982) preparatory to building a symbol-level model). In the COMMONKADS method (Breuker and Van de Velde, 1994), which inspires much of the work described in this paper, the intermediate model is termed the "expertise model". Advantages of an expertise model include:

- *Verification and Validation* (V&V): For a full discussion of the role of such an intermediate model in these important processes, see Vermesan and Bench-Capon (1995). Here, we will say only that the kind of V&V that can be performed on the actual representation reduces to a set of checks for internal coherence of the model; identifying redundant rules, dead-end rules, sets of conditions that cannot co-obtain in the model, and the like. While these "anomalies" can identify possible errors in the model, these errors must always to be referred back to the expert to see if there really is a problem. In the absence of an explicit intermediate model, therefore, appeal has to be made to the model implicit in the head of the expert. Since this is implicit, the model cannot be guaranteed to be itself consistent and coherent. Thus an explicit model is desirable as providing an objective standard against which the V&V activities can be performed.
- *Maintenance*: The case for an intermediate model to support this activity is made in works such as Coenen and Bench-Capon (1993) and Bratley et al. (1993). Essentially the argument here emphasises the particular importance of adaptive maintenance as against corrective and perfective maintenance for LKBS. This being so, it is typically changes to the source knowledge which require changes to the system, and thus it is essential that there is a very clear relationship between the sources and the code to be maintained. In the executable model it is often impossible to sustain this relationship. Moreover, when maintaining the system, the maintainer must have access to the design choices of the sort mentioned in the first paragraph of this paper, if the maintenance work is to be consistent with these choices. Again, these choices are better made explicitly available than inferred from the code itself.
- *Reuse*: This topic, discussed in Oskamp (1993), is the main concern of Visser (1995a). Since there is a very great investment in constructing a knowledge base for an LKBS it is important that as much mileage as possible be obtained from it. Thus, we want to reuse the knowledge base constructed for one application in another application in the same domain. With regard to executable code, this is often very difficult, since the knowledge will have been selected, represented and optimised for a particular task, assuming that it is to support that task as efficiently as possible. The aim in building the intermediate model is to provide a task-neutral representation, which can then be customised if necessary to support particular tasks, effectively separating the reusable knowledge from the task-specific knowledge.

Although now widely seen as desirable, there have been as yet few detailed attempts to build substantial models of this sort in the legal domain reported. Only through such experience will the problems and a method for attacking them

emerge. In order to make some experience available, in this paper we describe the construction of an expertise model for a particular piece of legislation; Articles 1 through 29 of the Dutch Unemployment Benefits Act (DUBA). In doing this we introduce a three-stage method found to be effective, and draw attention to some of the particular problems and issues that arose. It should be recognised that the purpose of following the method is not to build a system which has a functionality different in kind from what can be achieved by other development methods, or even using no development method. The aim is rather to improve the quality of the resulting software, so that it is more capable of verification and validation, more maintainable, and capable of reuse in other applications.

A key stage in the development of an expertise model is deciding on the entities and relations to be distinguished in the model. In general, such decisions can be made from different perspectives, resulting in different conceptualisations of the domain (and hence in different expertise models). Following Gruber (1992) we refer to an explicit conceptualisation of a domain as an ontology. The idea is to provide a knowledge level description of the terminology that eventually will be used to express the domain knowledge (Van Heijst, 1995). The method introduced in this paper can be seen as an ontological analysis in which we gradually work towards a (formal) ontology for the domain under consideration.

In the remainder of the paper, we first describe the domain (Section 2), then the method (Section 3). Next we discuss each stage of the method by applying it to the DUBA (Section 4), we provide a brief discussion on the ontology and its use (Section 5), and finally we make some concluding remarks (Section 6).

2. The Domain: Dutch Unemployment Benefits Act (DUBA)

The particular Act on which we focus is the Dutch Unemployment Benefits Act, as issued in November 1986. This legislation is thought to be representative of the sort of legislation for which LKBSs have been constructed (Van Kralingen 1995). It contains several different types of norms and acts, makes use of time related concepts, and has certain open-textured concepts, for instance, relating to the notion of "suitable employment".

The Act comprises eleven chapters, each divided into several parts, which divide into paragraphs, which in turn are divided into articles, which divide further into sections and subsections. Altogether there are 137 articles. Here we concentrate only on the first two chapters, and in particular articles 1 through 29. This part addresses three issues:

1. *Descriptions of legal concepts* (articles 1–14): these articles set out some of the central concepts used by the Act including the notions of *employee, relationship of employment, employer*, and *wage per day*.
2. *Requirements for entitlement to unemployment benefit* (articles 15–21): these articles set out the conditions that must be satisfied if unemployment benefit is to be awarded. Chief amongst these are that the employee must be unemployed,

and must have worked for 26 weeks in the last 12 months. Also contained here are certain exceptions to entitlement, and the conditions for determining when the entitlement starts and ends.

3. *Effectuation of entitlement* (articles 22–29): these articles set out what must be done if a person who is entitled to the benefit is actually to receive it. For example, the employee must not become culpably unemployed, and must be making a sufficient effort to obtain suitable employment. The articles also describe the sanctions that can be applied if these conditions are not met, including reducing the amount of the benefit, or withholding it in full.

The open-textured concept of *suitable employment* is central to the DUBA. It is important that we have an example of an open-textured concept since the treatment of open texture presents particular problems for an LKBS. Essentially there are three approaches: at one extreme the concept is treated as primitive and left for resolution to the unaided user, while at the other extreme some attempt is made to reason about the concept. The most usual course, however, is to attempt to supply some explication of the concept based on previous decisions and commentaries. This may involve providing sufficient conditions for its applicability and its non-applicability, or the identification of features which argue for and against its application. We adopt the last approach. The concept of suitable employment has been widely discussed, and now there is a reasonable consensus on its meaning. In particular de Wildt (1993) presents a model derived from previous decisions which can be used to simulate decisions, essentially based on the identification of six categories of factors, some pro and some con, which can be considered together to come to a decision. We follow this model in our work. Some factors are decisive: thus if the wage offered is below the minimum, the employment cannot be considered suitable, whereas others, such as the educational level of the employee, are only influences on the decision, and are assigned some kind of weight in the model, allowing them to be considered with other similar factors.

As can be seen from this brief description, the selected fragment of legislation is substantial, relatively complete in that it can form the basis of several tasks suitable for an LKBS, and representative of the sort of legislation which has formed the basis of many LKBS projects.

3. The Method

In building our model we drew principally on three different pieces of research; COMMONKADS (Aamodt et al., 1992; Wielinga et al., 1992), which supplies the overall idea of an expertise model and what it contains; a theory of frame-based conceptual models of statute law (Van Kralingen, 1995), used to supply a generic ontology of the legal domain; and KANT (Bench-Capon and Coenen, 1992), used to determine the vocabulary with which to instantiate the domain model. We discuss each of these strands in this section.

3.1. The Expertise Model of CommonKADS

The expertise model of CommonKADS consist of two parts: *application knowledge* and *problem-solving knowledge*. Below, we deal with each in turn.

3.1.1. *Application Knowledge*

A principal attraction of the CommonKADS expertise model is that it enforces a clear separation between control knowledge and domain knowledge, requiring independent descriptions of each. This is vital for our purposes since the compilation of domain and control knowledge is a major drawback of symbol-level models. The application knowledge is meant to include all the knowledge required for solving problems in the domain. Within this, three categories are distinguished:

1. Task Knowledge;
2. Inference Knowledge; and
3. Domain Knowledge.

Task knowledge, which together with the inference knowledge makes up the *control knowledge* of the expertise model, specifies how the problem-solving goals can be achieved. Tasks may be *primitive tasks*, or *composite tasks*, built up from primitive tasks. Primitive tasks are linked onto inferences. Each task has a *task definition* which specifies what the task can bring about, and a *task body* which describes the steps by which this can be achieved.

The inference knowledge specifies the primitive reasoning steps that can be performed. Each inference establishes a relation between entities drawn from the domain knowledge (see below). These are the only knowledge entities that can directly use the domain knowledge.

Domain knowledge comprises three components (Wielinga et al., 1992); a *domain ontology*; a *case model*; and the *domain models*. The domain ontology is an explicit conceptualisation of the domain (Gruber, 1992). It defines the vocabulary, typology and taxonomy with which the case and domain models can be specified. Within CommonKADS there is a conceptual modelling language, CML, which provides three constructs: objects (atomic and constructed), relations, and value sets.

The case model is a structured description of the dynamic data related to a case, and includes the *initial case data* (available before the reasoning commences) *intermediate data* (derived during the reasoning process) and the *case solution* (which is the ultimate result of the reasoning process). All of these are specified using the domain ontology.

The domain models are coherent sets of expressions, again specified using the domain ontology, which describe the domain from a particular perspective, such as its causal relations, or its components.

3.1.2. *Problem-Solving*

This part of the expertise model comprises problem-solving methods and strategic knowledge. Problem-solving methods specify the candidates which could be used to satisfy a certain class of task definitions. Strategic knowledge is used to select a relevant method from the available problem-solving methods. This part of the expertise model is less fully elaborated in the writings on COMMONKADS. Partly as a result of this lack of clear guidelines we did not, in fact, specify this part of the expertise model, and hence our expertise model of the DUBA consists only of the application knowledge: task, inference and domain knowledge.

3.2. VAN KRALINGEN'S CONCEPTUAL FRAME STRUCTURES

COMMONKADS is a domain-independent methodology, and in consequence, the method does not supply very much detailed support for particular domains. In particular the ontology needs to be extended for the domain under consideration. Some work has been done on this in law (Valente, 1995), but it is not sufficiently established to be immediately useful to us. We therefore required a set of domain specific structures for modelling the legal domain. For this we turned to Van Kralingen's work (1995).

Van Kralingen identifies three basic entities to be found in legal systems: *norms*, *acts* and *concept descriptions*. Each of these are provided with a frame structure specified at a conceptual level. Although frames have been used before in AI and Law, for example by Rissland and Ashley in their hypo system (Ashley, 1990), they have typically used them to model only cases. Van Kralingen's work offers the prospect of gaining the advantages of such a structured object representation for all the entities in the legal domain. Moreover, it seemed suitable for our purposes since it provides a means of structuring the domain in a way customised to law and it provides detailed schemes – firmly based on legal theory – of the entities relevant to our domain. The frame structures are useful both to guide the knowledge engineer in acquiring knowledge, and to provide a framework enabling the knowledge acquired to be assimilated. The slots of the frames identified show what information is required, and hence will identify potential gaps during the knowledge elicitation process, so as to drive further elicitation. When the knowledge is obtained, it is used to complete the identified structures. We will briefly discuss each of the three basic entities in turn.

3.2.1. *Norm Frames*

Norms are the general rules, standards and principles of behaviour that subjects of law are enjoined to comply with. They are statements to the effect that something ought to, may, or can be done. Norm frames comprise the following slots:

Norm identifier:	A point of reference for the norm.
Norm type:	Either norm of conduct or norm of competence.
Promulgation:	The source of the norm.
Scope:	The range of application of the norm.
Conditions of application:	The circumstances under which the norm is applicable.
Subject:	The person or persons to whom the norm is addressed.
Legal modality:	Either ought, ought not, may or can.
Act identifier:	A reference to a separate act description.

3.2.2. *Act Frames*

Acts represent the dynamic aspects which effect changes in the state of the world. There are two types of acts: events and processes. Events represent an instantaneous change between two states, while processes have duration. Act frames comprise the following slots:

Act identifier:	A point of reference for the act.
Promulgation:	The source of the act description.
Scope:	The range of application of the act description.
Agent:	Either an individual, a set of individuals, an aggregate or a conglomerate.
Act type:	Both basic acts and acts that have been specified elsewhere can be used.
Means:	Material objects used in the act or sub acts (e.g., a gun).
Manner:	The way in which objects have been used or sub acts have been performed (e.g., aggressively).
Temporal aspects:	An absolute time specification (e.g., on the first of August, on Sundays, at night, etc, but not: during a fire, after the King dies, etc).
Spatial aspects:	A specification of the location where the act takes place (e.g., in the Netherlands, in Leiden, on a train).
Circumstances:	A description of the circumstances under which the act takes place (e.g., during a war).
Cause:	A specification of the reason(s) to perform the action (e.g., revenge).
Aim:	The goal visualised by the agent (e.g., with a view to unlawfully appropriate an object).
Intentionality:	The state of mind of the agent (e.g., voluntary).
Final state:	The results and consequences of an action (e.g., the death of the victim).

3.2.3. *Concept Frames*

Concept frames deal with the meanings of the concepts found in the domain. They may be definitions or deeming provisions and can be used to determine definitively the meaning of a notion, either by, as in the case of the former, providing necessary and sufficient conditions, or, as in the case of the latter, establishing a legal fiction. Another type of concept is the factor, used to model open textured concepts. A factor may either establish a sufficient condition, or indicate some contribution to the applicability of the concept, as discussed in Section 2 above. Finally, we may have meta concepts which are provisions governing the application of other

provisions. Concept frames comprise the following slots:

Concept:	The concept to be described.
Concept type:	Either definition, deeming provision, factor or meta.
Priority:	The weight assigned to a factor.
Promulgation:	The source of the concept description.
Scope:	The range of application of the concept description.
Conditions:	The conditions under which a concept is applicable.
Instances:	An enumeration of instances of the concept.

3.3. INSTANTIATING THE DOMAIN MODELS USING KANT

Van Kralingen (1995) provides us with an explicit conceptualisation of the domain, which we use as the basis of our domain ontology. What is now required is a way of instantiating these frame structures so as to produce the domain models. To do this we must identify the specific vocabulary for our detailed domain: the predicates and constants that we will use. The method used to achieve this is based on the Knowledge ANalysis Tool (KANT) as described by Bench-Capon and Coenen (1992).

KANT provides an analysis method to move from a (legal) document to a hierarchy of classes, identifying the attributes for each class and the values that each attribute can take. The method proceeds through the construction of a series of structured English notes each recording a step in the analysis. These notes are termed "Freestyle" structures.

The first step is to identify the elements of interest in the domain. In particular, the analyst passes through the source documents marking the entities found in them, the attributes ascribed to those entities and any tests that are applied to them. Tests, such as "is age greater than 60?" are valuable indications of the values that are given to the attributes. The attributes and tests for each entity are collected into a Freestyle structure, the "Tests-On-Objects" (TOO) structure. Therewith, we effectively produce a list of the various data, initial, intermediate and final, which are *required* by the case model.

The next step comprises the creation of the Entity-Attribute-Value structure (or EAV structure), which is a set of the Entity-Attribute-Value triples. To this end, the analyst collects the possible values that the attributes (of the entities) can take. The EAV triples represent the expressions that are required to specify the rules and data of the domain.

The following step is to organise the EAV triples into a class hierarchy. The principles governing the organisation is that a sub-class must have all the attributes of its super-class, and may have additional attributes, and that a range of values for an attribute can be contracted in a sub-class but not expanded. This ensures that sub-classes are *strict* specialisations of their super-class. In this step we also finalise the attributes that entities can have, and the possible values they can take, by bringing what we have produced into line with the above principles, and sometimes

by seeking external information. This is important both as a way of tightening up on the definitions that we use, and of identifying some implicit knowledge. For instance, the only explicit test on the attribute "sex" to be found in the legislation may be to see if someone is female. It may, however, be important to record that this attribute can also take the value "male", and no other value; or that it can take the values "male" or "neuter" and no others. One of these choices will be appropriate in the context, and needs to be recorded. Similarly if we have the tests "is age less than 18?", "is age greater than 60?" and "is age greater than 65?" we could say that age is an integer in the range at least 17 to 66. We need, however, to find out what the appropriate range really is: in this case 0 to 150 may be chosen. The choice here does however have implications, for example in the validation of data, and in the consequences that can be drawn from a negative answer to "is age less than 18?".

At this point we will have a tightly defined vocabulary for the domain which will enable us to instantiate the frame structures. The method ensures that the vocabulary contains only those entities and their attributes that play a role in the sources from which they have been derived, and that their values are crisply defined so as to allow for the expression of the predications found in the sources, and so as to be free from any implicit assumptions as to what the possible values for these attributes might be.

4. Applying the Method to the DUBA

In this section we discuss how the method described above was applied to the DUBA. We describe the process by considering each of the three techniques introduced. As we go we will highlight the particular problems and issues that arose, and say how they were resolved. This section will contain a high degree of detail. we believe this is necessary both to give an impression of the depth to which an analysis must go, and because it is at the most detailed level that the sharpest tests for the method are presented.

4.1. Identification of Domain Knowledge

The key first stage is to identify the knowledge that we wish to include as domain knowledge and to distinguish it from the knowledge we wish to include as control knowledge. In particular we must examine the legal knowledge contained in our chosen fragment of the DUBA to consider how it is best apportioned between these two knowledge types.

An obvious answer, traditional when modelling legislation, is to consider all the information in the legislation to be domain knowledge: after all the legislation is, in some sense, the domain of the application. Closer inspection of what is actually done, however, reveals that matters are less simple. In the seminal paper on the formalisation of the British Nationality Act (Sergot et al., 1986), not all the legislation was in fact formalised. The parts not formalised:

> "consist mostly of descriptions of amendments to other Acts, repeals, offenses and proceedings related to them, and decisions involving exercise of description by the Secretary of State".

Thus even in this work, which set out to be a formalisation of the Act, some of the legislation is found not to require inclusion as domain knowledge. If we are to propose a systematic methodological approach to formalising legislation, however, it is necessary to characterise the legislation that should be excluded from the domain knowledge, so that its inclusion or exclusion is not simply a matter of judgement by the modeller. Even more important is that we identify knowledge that is properly excluded from the domain knowledge, but which requires inclusion as control knowledge.

In what follows we argue that the legislation does contain information best regarded as control knowledge, and how this may be recognised.

4.1.1. *Defining Control Knowledge*

Solving a problem is performed by a series of problem-solving actions (e.g., invoking a procedure, deriving new information from the knowledge base, acquiring data). In general, at each point in the problem-solving process a choice of problem-solving actions is available. Following Davis (1980), we distinguish three phases with respect to the invocation of a (problem-solving) action*:

1. *identification of candidate actions* (collecting the set of actions that can be performed);
2. *conflict resolution* (selecting an action from the set of possible actions);
3. *execution* (the execution of the selected action).

The problem of selecting an action in the conflict-resolution phase is called the control problem (cf. Hayes-Roth, 1988, p. 505). Providing a strategy for making a selection is referred to as solving the control problem; the knowledge used to solve the control problem is called control knowledge (Clancey, 1986, p. 4). We note, that if the action can be executed by the system itself (instead of by other agents), then control knowledge is an instance of meta-knowledge. As a consequence, reasoning about such control problems is meta-level reasoning (cf. Davis, 1980).

To identify control knowledge in legal texts, we have to identify problem-solving actions. Moreover, there should be knowledge about selecting a problem-solving action from a set of possible problem-solving actions. Following Franken (1988) we assume legal texts to consist of legal rules (legal rules can be any kind of text fragment, such as norms, articles, sections, concept descriptions). A problem-solving action is assumed to be the application of such a legal rule (e.g., drawing a conclusion using this legal rule), or the examination of a set of legal rules. We can now state that the identification of control knowledge in legal texts involves

* We use a slightly different terminology. In particular, we avoid the term knowledge source because this term was also used to denote primitive inferences in KADS-I. Davis (1980) uses the term in a more general way (e.g., procedure, inference rule, theorem).

identifying legal rules as well as the knowledge about selecting a legal rule from a set of legal rules. In other words, we focus on those control problems in which there is a choice between the application of different legal rules.

4.1.2. *Control Knowledge in Legal Texts*

Our discussion of control knowledge in legal texts concentrates on what we propose to call *normative control knowledge*, denoting legal knowledge which in some way obligates the user of the legal text (e.g., a judge) to solve control problems in a predefined way. For instance, to sentence a suspect, it is required that the charge is proven (obligating the judge first to determine whether the charge is proven, and then to find an appropriate sentence). We do not discuss forms of control knowledge in which the user of the text may choose how to solve the control problem without there being a legal obligation to do it in that particular way (for such control problems see, for instance, Van Kralingen and Visser, 1991; Groendijk and Oskamp, 1993). Below, we discuss two types of normative control knowledge in legal texts, namely control knowledge in: (1) *procedural norms of competence*, and (2) *conflict-resolution knowledge*.

4.1.3. *Procedural Norms of Competence*

The first type of control knowledge in legal texts is contained in procedural norms of competence (Van Kralingen, 1995). These norms state such things as who is empowered to perform certain actions, which preconditions must be satisfied before an action can be applied, and the like.

We note that norms guiding the behaviour of people in society have different normative functions (e.g., Hart, 1961; Brouwer, 1990) and that one of these normative functions deals with the control of the creation and application of other legal rules. These norms are norms of competence; they detail the manner and form in which the conferred power is to be exercised (Hart, 1961, p. 28; Van Kralingen, 1995). The procedural norm of competence (henceforth: PNC) is a special kind of these norms, namely, the norms of competence which control the creation and the application of other legal rules by prescribing legal procedures. Some of these procedural norms of competence comprise control knowledge. This is argued below.

A PNC defines one or more intermediate steps which a norm subject has to take in order to reach a certain legal goal. Possibly, one or more of the intermediate steps can be achieved only through the application of (other) legal rules (e.g., deriving intermediate legal conclusions). This possibility leads to the distinction between:

(a) *object-level PNCs*, and
(b) *meta-level PNCs*.

Below, we define these types of norms and illustrate that the second type of norm, in contrast to the first type of norm, comprises control knowledge.

(a) *object-level PNCs* are PNCs that can be performed without invoking other legal rules. For instance, when the Social Insurance Board wants to change some directives, then according to article 125 of the DUBA, an intermediate step should be the announcement of the change in an official newspaper. Although these object-level norms describe a flow of control through possible actions, the actions do not involve the invocation of legal rules. In other words, the set of possible problem-solving actions does not contain an action that requires the invocation of a legal rule. Therefore, this kind of PNC does not comprise control knowledge according to our definition, since the satisfaction of the precondition can be recorded as a fact, rather than needing to be deduced from the other legal rules (and so, they say nothing about the order in which other legal rules should be applied).

(b) *meta-level PNCs* are PNCs that do require other legal rules to be invoked. An example can be found in the Dutch law of criminal procedure. In article 350 of this statute it is stated that the following succession of questions needs to be answered before a person can be sentenced:

(350-1) are the facts described in the charge proved?
(350-2) do the facts described in the charge yield an offence?
(350-3) is the offender punishable? and
(350-4) what punishment or measure has to be imposed?

As will be clear some of these intermediate steps require other legal rules to be applied (e.g., 350-2 requires examination of the offenses described in the Dutch Criminal Laws). Because the problem-solving action that invokes article 350 involves invoking other legal rules, the article is considered to contain control knowledge.

If we make an expertise model of a statute with PNCs, then these norms should be specified. Object-level PNCs do not comprise control knowledge, and, hence, can be specified as domain knowledge. With respect to the meta-level PNCs we are faced with a dilemma. On the one hand, norms derived from legal texts must be specified as domain knowledge. On the other hand, these norms comprise control knowledge, since they impose an order in which things must be done, and should therefore be specified as control knowledge (in terms of a COMMONKADS expertise model). Note, that whenever a meta-level PNC is specified as control knowledge, the norm cannot be breached by the system. The system just enforces the norms by performing the prescribed problem-solving actions. This can both be a desirable feature (e.g., in case the system has to determine one's entitlement) as well as a non-desirable feature (e.g., in case the system should be able to monitor whether an institution complies with legal procedures).

Our dilemma has a pragmatic solution with respect to the DUBA in that meta-level PNCs do not occur in this statute. We note that the DUBA does contain several

PNCs, for instance, in articles 111 through 127. All these norms are in fact object level PNCs because the intermediate steps defined in these norms do not require other rules to be invoked (that is, not in our fragment of the DUBA). Hence, we find a practical solution for our dilemma: our legal texts do not contain control knowledge in the form of (meta-level) PNCs. However, in the general case, the question will need resolution, since this pragmatic solution will not always be available.

4.1.4. *Conflict-Resolution Knowledge*

The second form of control knowledge in legal texts is found in what we refer to as (legal) conflict-resolution knowledge. By this we mean the legal knowledge used to discriminate between conflicting conclusions derived from different legal rules. Here, we illustrate that with respect to this knowledge the same dilemma arises as with procedural norms of competence. Although the knowledge is for a large part contained in legal texts it is to be considered control knowledge according to the definition given above.

We discuss the dilemma by first defining more precisely what is meant by a conflict and by conflict-resolution knowledge. After this, we illustrate that conflict-resolution knowledge is contained in legal texts. In particular, we show the potential conflicts that may arise from the DUBA and the knowledge used to solve these conflicts. This leads to a distinction of conflict-resolution knowledge into two parts. Then, we show that conflict-resolution knowledge is an instance of control knowledge, thus yielding a dilemma. Finally, we resolve the dilemma by choosing what part of the conflict-resolution knowledge is modelled as domain knowledge and what part is modelled as control knowledge in the expertise model.

4.1.4.1. *Defining a Conflict and Conflict-resolution Knowledge*

A common technique in drafting legislation is to state a general rule, and then refine the rule by stating some exceptions which lead to a different conclusion. Were both the general rule and the specific rule to be applied, a contradiction would result. In fact, no contradiction arises in practice, since only one of the rules will be applied. However, the potential for a conflict arises and we need conflict-resolution knowledge to determine which rule is to be applied, so that the conflict can be avoided.

Here, we define a conflict as a state in which there are two conclusions which, from a legal point of view, cannot be or remain true during the rest of the reasoning process (cf. Aarnio, 1987, p. 113). According to this definition, the conclusion from a rule, and the conclusion (from another rule) that the former rule is not applicable is considered a conflict. Also, there is a conflict if two rules give incompatible values for an attribute, e.g., 'twelve years of imprisonment' and 'fifteen years of imprisonment' (example from Prakken, 1993, p. 30).

Conflicts can be dealt with by preferring one of the conclusions. This requires knowledge on the preference relations between the conclusions (or on the rules

that were used to derive the conclusions). This knowledge is, for instance, required to determine that if there is a conclusion that a rule is not applicable then this conclusion is to be preferred over the conclusion of that rule. The knowledge that is used to solve conflicts, that is, to prefer one of the conclusions, is here referred to as conflict-resolution knowledge.

Conflicts and Conflict-Resolution Knowledge in the DUBA

Legal texts are stated in such a way that conflicts (as we defined them) frequently occur (cf. Thornton, 1987). Because most of these conflicts are anticipated by the drafter of the legal text (e.g., exceptions to rules) there apparently must be conflict-resolution knowledge available in some form. So far, we did not elaborate on the appearance of this conflict-resolution knowledge in the legal texts. Below, we illustrate that conflict-resolution knowledge is partly contained in legal texts. We examine the DUBA for (potential) conflicts and discuss the related conflict-resolution knowledge. We do not consider conflicts that can arise as a result of the introduction of new (fragments) of ruling (such as conflicts between older and newer versions of legal rules).

To identify conflicts in the DUBA we use the definition given above. We identify pairs of legal rules of which their conclusions cannot be or remain true from a legal point of view. These pairs are listed below. For reasons of brevity we list only one example of pairs of legal rules that cause each of the conflicts (other pairs of rules that yield the same conflicting conclusions are left out). This yields the following six cases (we assume these conflicts are the only conflicts that can occur):

(c1) conclusion 'employee' (derived, for instance, from article 3 section 1) and conclusion 'not an employee' (derived, for instance, from article 3, section 2);
(c2) conclusion 'relationship of employment' (derived from article 4 section 1) and conclusion 'article 4 section 1 is not applicable' (derived from article 4 section 2);
(c3) conclusion 'relationship of employment' (derived, for instance, from article 4) and conclusion 'no relationship of employment' (derived, for instance, from article 6);
(c4) conclusion 'entitled to benefit' (derived from article 17) and conclusion 'article 17 is not applicable' (derived from article 18);
(c5) conclusion 'entitled to benefit (derived, for instance, from article 17) and conclusion 'not entitled to benefit' (derived from, for instance, article 19);
(c6) conclusion 'suitable employment' (derived from, for instance, an employees employment history (as meant in De Wildt 1993, p. 37), and conclusion 'not suitable employment' (derived from, for instance, the status of the work offered as meant in De Wildt 1993, p. 37).

The conflict-resolution knowledge with respect to these conflicts is partly contained in the legal texts. We illustrate this. If we study the conflicts listed we can distinguish different kinds of conflict-resolution knowledge. In conflicts (c2) and (c4) the rules involved in the conflict explicitly state which of the two rules

involved in a conflict is preferred; the one rule explicitly states the other rule not to be applicable. Obviously, this information is contained in the legal text. Beside this information, the conflict-resolution knowledge comprises a principle stating that the conclusion of the inapplicable rule should be withdrawn, unless it can be justified by some other, applicable, rule. In this example, we consider the conflict-resolution knowledge to comprise two parts, namely (a) the information that a certain rule renders another rule inapplicable, and (b) a general principle stating that a rule that renders another rule inapplicable should be preferred over the rule that is rendered inapplicable.

With respect to conflicts (c1), (c3) and (c5) we find no explicit statement about how the conflicts have to be solved. This does not imply that there is no solution to these conflicts (they are not considered to be undecided conflicts, see Prakken, 1993). Because the conflicts arise between a general rule and an exception there is an unequivocal solution to the conflict: when both legal rules are applicable the exception is preferred over the general rule. Which legal rule is the exception and which legal rule is the general rule is not stated explicitly, but it is assumed that the user of the text is able to derive this information from the legal text together with knowledge of the structure of legal texts, common sense, and general understanding of natural language that enables one of the rules to be recognised as the more specific rule. For example, in legal texts, exceptions usually are stated directly after the general rule. Here, the conflict-resolution knowledge is considered to comprise: (a) the information that a certain rule is a general rule, and that another rule is an exception to that rule, and (b) a general principle that states that the conclusion of the exception should be preferred over the conclusion of the general rule (the well-known principle Lex Specialis Derogat Legi Generali).

Conflict (c6) differs from the other conflicts in that the legal rules involved in the conflict do not provide explicit or implicit information about how to solve the conflict. This is typical of an open textured concept where there are features that argue both for an against the application of the concept. This is because here the legal rules do not stand on their own, they cannot be viewed in isolation of all other legal rules that relate to the same concept. We note that this contrasts with the conflicts (c1) through (c5) which can be solved without considering other legal rules. A conclusion on the notion of suitability only *contributes to* the conclusion on the conflict. The final conclusion is drawn on the basis of the sum of all the weights of conclusions pro and contra suitability. The weight of each individual conclusion depends on the weight assigned in the concept description. This information is given in the model in the commentary we follow (De Wildt, 1993). Again, the conflict-resolution knowledge comprises two parts: (a) the weights of the factors involved in the conflict, and (b) a general principle stating that the conclusion with the highest total weight should be preferred over the other conclusion.

In summary, we observe that in all the conflicts, (c1) through (c6), the conflict-resolution knowledge comprises two parts: (a) some information about the rules involved in the conflict, and (b) a general principle that applies this information

to prefer one of the rules (or conclusions) in the conflict. Moreover, we note that the first part of the conflict-resolution knowledge, the information about the rules involved in the conflict, is either contained in legal texts, or can directly be derived from the texts. In contrast, the second part of the conflict-resolution knowledge, the general principles that are applied to solve the conflicts, is not specified with the legal text. These principles are implicit and assumed by the drafter to be known by whoever is interpreting the legislation.

4.1.4.2. *Conflict-Resolution Knowledge Is Control Knowledge*

We have illustrated that conflict-resolution knowledge is partly contained in legal texts. Here, we argue that this conflict-resolution knowledge is an instance of control knowledge. To see this, we reconsider our definitions of control knowledge and conflicts. Assume we have two problem-solving actions a1 and a2. Problem-solving action a1 is the application of a rule r1 and a2 is the application of a rule r2. The conclusions of these rules, conclusions c1 and c2, are in conflict. Because only one of these conclusions can be preserved, one conclusion has to be preferred. This preferred conclusion is used for making further inferences during the problem-solving process. From a different perspective, we may state that only one of the rules r1 or r2 can be applied successfully. If so, the process of preferring one conclusion is an instance of the control problem in which the set of possible actions is $\{a1, a2\}$. For this reason, we consider the problem of preferring a conclusion as a control problem, and the knowledge used in this process as control knowledge (cf. Tan and Treur, 1991).

The conclusions derived pose the same dilemma as discussed before. On the one hand, we illustrated that conflict-resolution knowledge is partly contained in legal texts, and this suggests that we should model this part of the conflict-resolution knowledge as domain knowledge (in the expertise model). On the other hand, we concluded that conflict resolution knowledge is control knowledge, which suggests that we should model this knowledge as control knowledge (in the expertise model).

To cope with the dilemma, we have to decide upon the question how to specify conflict-resolution knowledge. Roughly, there are three options to deal with conflicts in a formal specification*:

1. *compiling out conflicts*: here the control knowledge is incorporated in the body of the rule. For example if r1 is preferred to r2 then that the conditions of r1 do not hold are added as conditions in r2. That this is undesirable is shown in Routen and Bench-Capon (1991): the problem is that the resulting lack of correspondence between the structure of the rules and their sources make maintenance and validation problematic.
2. *Making the inference engine deal with conflicts*: this is the approach taken in classic production rule systems such as those implemented in OPS5 (Brownston et al., 1985). Some general conflict-resolution principle such as specificity or

* In Prakken (1993) a more elaborate discussion of options is provided.

recency is chosen and then uniformly applied by the rule interpreter. The problem with this is its lack of flexibility – the same principle is always applied in the same way. Where the principle is not appropriate it may become necessary to distort the rules themselves. For example if specificity is being used it may be necessary to add bogus conditions to a rule to ensure that it is more specific than its competitors.

3. *Performing explicit meta-level reasoning about conflicts*: here the principles themselves are stated explicitly and are themselves rules that are used to determine the applicability of the object level rules. This approach is well described in Sartor (1992). The problem arises when the principles themselves conflict, whereupon we must once again choose between these three approaches for resolving conflicts at the meta-level.

We choose a solution which draws elements from both the second and the third options. From the conflict-resolution knowledge, we specify the general principles as control knowledge (in the expertise model), so that the inference engine is customised to apply the general legal principles, and the information about the rules as domain knowledge (also in the expertise model), so that where conflicts are explicitly resolved in the statute, the knowledge used to resolve them is explicitly represented also. Thus, information such as explicit statements concerning the applicability, weights and promulgations are all specified as domain knowledge. The way this information is used is explicitly modelled (in task specifications) as control knowledge, while the information used by these rule-selecting tasks is modelled as domain knowledge.

4.1.5. *Identification of Structured Entities: Norms, Acts and Concepts*

The domain knowledge in an expertise model is structured as a set of domain models (cf. Breuker and Van de Velde, 1994, p. 22). The most important domain models in our domain specification represent the norms, acts and concept descriptions in the legal domain. In this subsection, we elaborate on these structured entities in the DUBA. We focus on the conceptual framework as defined by Van Kralingen (1995) and identify what frames have to be created for norms, acts, and concept descriptions, respectively.

4.1.5.1. *Norms*

A norm is a statement to the effect that some action ought, ought not, must, or can be done. We already stated that the DUBA comprises norms of different nature (e.g., norms of competence, and norms of conduct). We here elaborate on the norms in the DUBA.

In our expertise model we recorded all the norms found in the chosen fragment of the DUBA. However, when building an application, we must make a selection of these norms, since some of the norms are irrelevant with respect to the problem-solving tasks we selected for implementation (assessment and planning to receive

the benefit). Most of the norms considered irrelevant deal with amendments. In building the system we assume the 'legal contents' of the DUBA to be static (no amendments are made to the statute during the problem-solving tasks, although of course the system will require maintenance to keep the knowledge up to date as the legislation changes). Considering the DUBA several norms concern amendments. These norms deal with the way (new or more detailed) regulations have to be issued, and with exceptional circumstances in which some agents (e.g., a Minister) may overrule the existing regulations in the statute. Article 12 section 2, for instance, states that the Minister of Social Services and Employment may, in contrast to what is stated in articles 9 and 10, appoint someone as an employer. Assuming the 'legal contents' of the statute to be static, implies that all norms that state how amendments should be issued are not relevant for our specification.*

We note that most norms that deal with amendments can be recognised by their norm subject. This is because making amendments to regulations is reserved for specific agents. In the DUBA we distinguish ten agents who are subject to a norm:

1. the employee,
2. the employer,
3. the industrial insurance board,
4. the Minister of Social Services and Employment,
5. the Minister of Defence,
6. the Minister of the Interior,
7. the Minister of Education and Science,
8. the Social Insurance Board,
9. employment offices, and
10. the General Unemployment Fund.

Six of these agents (agents (4) through (8) and (10)) only play a role in norms that deal with amendments. As a consequence the set of relevant norms considered here is restricted to those directed at the agents employee, employer, the industrial insurance board and employment offices. From the norms that remain we select 12 norms which we assume to be representative for the DUBA. These norms are directed at the industrial insurance board and the employee.

Norms with the industrial insurance board as norm subject:

(n1) the industrial insurance board can reduce a benefit if the employee is not fully available to perform labour or if the employee performs new labour since his benefit has been determined (article 20);

(n2) the industrial insurance board can terminate a benefit if the employee is not fully available to perform labour or if the employee performs new labour since his benefit has been determined (article 20),

* Although we assume the legal contents to be static, at any given time the Minister will have appointed some set of people as employers. This is specified using extensional concept descriptions (see below).

(n3) the industrial insurance board ought to supply a benefit whenever an employee is entitled (article 22);

(n4) the industrial insurance board ought to notify the employee of their decision with respect to the benefit (article 22);

(n5) the industrial insurance board can reduce a benefit when the employee does not meet his or her obligations laid down in articles 24 and 26 (article 27);

(n6) the industrial insurance board can terminate a benefit when the employee does not meet his or her obligations laid down in articles 24 and 26 (article 27);

(n7) the industrial insurance board can temporarily reduce a benefit when the employee does not meet his or her obligations laid down in articles 24 and 26 (article 27);

(n8) the industrial insurance board can temporarily terminate a benefit when the employee does not meet his or her obligations laid down in articles 24 and 26 (article 27);

Norms with the employee as norm subject:

(n9) the employee ought not become culpably unemployed (article 24);

(n10) the employee ought to accept suitable employment (article 24);

(n11) the employee ought to submit a request in order to obtain benefits (article 26, section 1, paragraph b);

(n12) the employee ought to register at the industrial insurance board and keep this registration up to date (article 26, section 1, paragraph d).

4.1.5.2. *Acts*

A legal system guides the behaviour of persons and institutions in society. This behaviour is expressed by means of acts that are forbidden, permitted, and obligated. In this subsection we discuss the acts in the DUBA that will be specified in the domain specification.

Acts are specific instances of the more general category of occurrences, the latter being the category of all things that occur in the world (Allen, 1984, p. 123). In the DUBA different occurrences can be distinguished, all of which can be shown to play a role in the entitlement of an employee to unemployment benefits. Before we list the occurrences in the DUBA we discuss six categories of occurrences. In particular, we distinguish between (a) acts and non-acts, (b) events and processes, and (c) institutional and physical acts.

(a) *acts and non-acts*

Occurrences can be initiated by agents. An example is the submission of a request for unemployment benefits. Also, there are occurrences that are (usually) not initiated by agents, such as snow fall or high water. We here define an act as an

occurrence initiated by a human agent (Georgeff, 1987). Occurrences not initiated by agents are defined as non-acts.*

We note that legal texts mainly deal with acts because legal systems assign a normative status to occurrences initiated by persons or institutions. Despite the emphasis on acts in legal texts, non-acts also play a role. An employee who is unemployed solely because of high water, for instance, can be entitled to unemployment benefits (article 18).

(b) *events and processes*

Following many authors, we partition the category of occurrences into events and processes (e.g., Allen, 1984, p. 132). Events are occurrences that are assumed to occur at a single point of time. For instance, we here assume the submission of a request for a benefit to be an event. Processes are occurrences that extend over a period of time, and so have a duration. Accordingly, processes have both a point of time at which they start, and a later point of time at which they end. With respect to processes we make two assumptions. The first assumption is that the duration of a process and therefore the point of time at which the process ends is known at the start of the process. If the duration of an occurrence is not known (e.g., rain), we consider it to be composed of two separate events (e.g., it starts raining, and it stops raining). The second assumption is that processes only have an effect on the world during the process, and not after the process. Otherwise stated, a process changes the world from an old state into a new state when it starts, and changes it from the new state back into the old state when it ends (the assumption allows us to describe the end of a process as the inverse of the start of a process).

An example of a process is the supply of a benefit. In the domain specification we specify events and processes in two different formal frames (instead of using one formal act frame to cover both). These formal frames are called event frames and process frames. The frames differ in the naming of their slots and in the way they are used during reasoning; unless it is confusing we use act frames to denote both types of frames.

(c) *institutional and physical acts*

Within the category of acts we distinguish institutional and physical acts (cf. Ruiter, 1993, p. 2). The subdivision relates to the distinction between (real-)world knowledge and legal (or regulation) knowledge (Gardner, 1987; Breuker and den Haan, 1991; Ruiter, 1993). A physical act is an act that can be performed in the world. An institutional act is an act that cannot directly be performed in the world, since it is an institutional interpretation of a physical act. Descriptions of institutional acts are found in legal texts. By mapping physical acts onto institutional

* In many articles on planning a distinction is made between events and acts (e.g., Georgeff, 1987, p. 6; Tate et al., 1990, p. 26). The latter being occurrences initiated by agents, whereas the former are not initiated by agents. Here, we use the term non-acts for occurrences not initiated by agents, and assume that events can be initiated by agents.

acts, a legal status can be given to the physical act (cf. Gardner, 1987). For instance, if an employee accepts a labour relation (a physical act), then this act may be interpreted as accepting a suitable labour offer (an institutional act), or, the act may be interpreted as accepting a non-suitable labour offer (also an institutional act). We note that the distinction between institutional and physical acts is important in planning tasks. Because an agent is not capable of directly performing institutional acts, we want to be able to propose a series of physical acts which can realise an institutional act.

An implication of the distinction between physical and institutional acts is that norms only refer to institutional acts. Below, we list some examples of institutional events and processes that are contained in the domain specification (we specify the norms that refer to the act in brackets using the numbers given in Section 4.1.5.1):

Institutional events:

(ie1) the employee accepting suitable employment (n10);
(ie2) the employee avoiding becoming culpably unemployed (n9);
(ie3) the employee submitting a request for benefit (n11);
(ie4) the industrial insurance board notifying the employee of a benefit supply or refusal (n4);
(ie5) the industrial insurance board terminating a benefit (n2 and n6).

Institutional processes:

(ip1) the employee registering at the employment office and keeping the registration up to date (n12);
(ip2) the industrial insurance board reducing a benefit because the employee is not (any longer) available for the labour market or if the employee performs new labour (n1);
(ip3) the industrial insurance board reducing a benefit because the employee does not satisfy the obligations laid down in articles 24 and 26 (n5);
(ip4) the industrial insurance board supplying a benefit (n3);
(ip5) the industrial insurance board temporarily reducing a benefit because the employee does not satisfy the obligations laid down in articles 24 and 26 (n7);
(ip6) the industrial insurance board temporarily terminating a benefit because the employee does not satisfy the obligations laid down in articles 24 and 26 (n8).

To be entitled to a benefit an employee has to make himself available for the labour market. Making oneself available for the labour market is not explicitly obligated for an employee. Availability for the labour market is a necessary condition for being unemployed, and therefore, for being entitled (article 16). Thus, although the act of making oneself available is not explicitly obligated we cannot omit the act from our domain specification (since it is required for the planning task; to be able to propose a set of actions that make an employee entitled). Other

acts are represented in the domain specification that are not explicitly forbidden, permitted, or obligated. Although these acts are not (directly or indirectly) subject to a norm, they may influence entitlement. An instance of such an act is moving to another town (which may influence travelling distances which in turn may cause a job offer to be not suitable). Below, we list some examples of physical events and processes that are contained in the domain specification.

Physical events:
(pe1) a person accepting a labour relation;
(pe2) a person making him/herself available for the labour market;
(pe3) a person increasing his/her availability for the labour market;
(pe4) a person resigning voluntarily;
(pe5) a person moving to another city;
(pe6) a person submitting a request for benefit;
(pe7) a person offering a labour relation;
(pe8) a person dismissing another person after the termination of a labour relation;
(pe9) an institution notifying a person of a benefit refusal;
(pe10) an institution terminating a benefit.

Physical processes:
(pp1) an institution reducing a benefit because a person is not (any longer) available for the labour market or if this person performs new labour;
(pp2) an institution reducing a benefit because a person does not satisfy the obligations laid down in articles 24 and 26 (we note that this reduction involves a different sanction from that mentioned in pp1);
(pp3) a person registering with an institution so as to keep his registration up to date;
(pp4) an institution supplying a benefit;
(pp5) an institution temporarily reducing a benefit;
(pp6) an institution temporarily terminating a benefit.

Some acts occur both as a physical and as an institutional act (e.g., processes (ip2) and (pp1) in which the industrial insurance board reduces a benefit). In such acts the physical-world (or: common sense) interpretation of the act closely resembles (or equals) its institutional interpretation. Although both acts have the same act specifications, we need to both acts; one for generating plans that can be performed in the world, and one for assessment of the legal status of a real-world act.

4.1.5.3. *Concept Descriptions*

Concept descriptions lay down the meaning of legal (or institutional) notions. In our domain specification we distinguish three types of concept descriptions (Van

Kralingen, 1995)*: (a) *definitions*, (b) *deeming provisions*, and (c) *factors*. Below, we discuss the concept descriptions in the DUBA.

(a) *definitions*. A definition is a concept description that states the necessary and sufficient conditions to determine whether a problem case is an instance of the concept described. Definitions are extensional if the conditions consist of a list of all the instances of the concept, and intensional if the conditions are abstract (cf. Franken, 1987). In the domain specification, we only have intensional definitions. We distinguish seventeen concepts described through definitions. These concepts are:

(df1) (legal) body,
(df2) continued payment of wages,
(df3) culpably unemployed,
(df4) employee,
(df5) employer,
(df6) entitled to benefits,
(df7) exceptional natural circumstances,
(df8) general unemployment fund,
(df9) industrial insurance board,
(df10) labour volume as defined in article 16,
(df11) minimum wage,
(df12) the Minister,
(df13) private-law labour relationship,
(df14) public-law labour relationship,
(df15) reduced payment fund,
(df16) unemployed, and
(df17) wage.

We note that some concepts (e.g., the concept employee) have their definitions extended by deeming provisions, which will be discussed next.

(b) *deeming provisions*. A deeming provision is a concept description that states the necessary and sufficient conditions to determine whether a problem case is an instance of the concept described, notwithstanding the fact that it does not fall under the definition of that concept. This gives the concept a new interpretation (creating a legal fiction). The difference between definitions and deeming provisions is that a deeming provision is a modification of an existing interpretation of the concept described (e.g., a common sense interpretation, or a definition in another act). To cite Routen and Bench-Capon (1991, p. 78): deeming provisions "allow things which are not true to be treated as if they were". Instances that normally do

* We do not use meta concepts to deal with textual constructions. Applicability restrictions are specified as (exceptions to) definitions.

not satisfy a certain concept interpretation are explicitly stated to satisfy the new concept interpretation (or vice versa). As an example, Routen and Bench-Capon mention a provision that states that a person who has regained eyesight within the previous six months shall be treated as if he were blind. The deeming provision here relies on an existing (and external) interpretation of the concept blind.

How can deeming provisions in the DUBA be recognised? In the example of Routen and Bench-Capon, it is obvious that according to a common sense interpretation of the concept blind, someone who has regained eyesight is not blind any more. Here, the deeming provision states that a person (who is not blind) has to be considered blind. However, recognising a deeming provision is not always obvious because the already existing interpretation of the concept may be unclear. In article 3 section 2 of the DUBA it is stated that someone who fulfils his employer-employee relationship outside the Netherlands shall not be considered an employee. According to Andringa et al. (1987) this statement can be seen as a deeming provision. In the example the interpretation of the concept employee does not offer much help because it is not obvious that normally, a person fulfilling an employer-employee relationship outside the Netherlands is an employee. This sort of provision is made to ensure that an expected interpretation is made. An alternative and less subjective way to recognise deeming provisions is by means of textual constructions. For instance, by statements as "... shall also be regarded as ...", "... shall be treated as ...", "... is also considered a ...", or "... is deemed ...". Here, we adopt such a criterion: deeming provisions are recognised by textual constructions such as the ones above.

If we examine the DUBA for such textual constructions, we find that four concepts are defined using deeming provisions. Below, we list these:

(dp1) employee,
(dp2) employer,
(dp3) suitable employment, and
(dp4) relationship of employment.

Most of these concepts are also described by means of definitions. Usually, the definition provides the first interpretation of the concept after which the deeming provision then modifies this interpretation. We note that there is only one concept that is described solely by a deeming provision (in the DUBA). This is the concept "relationship of employment". In fact, the definition of this concept is found in another statute (1637BW). By the deeming provision in the DUBA the concept is given a slightly different interpretation: some labour relations also qualify as a relationship of employment (e.g., article 4), other labour relations that in 1637BW qualify as a relationship of employment do not quality as such in the DUBA (article 6).

(c) *factors*. A factor is a concept description used to describe open-textured concepts, stating the conditions that must be considered when applying such an open-textured concept. Satisfying the conditions of a factor is not a guarantee for the applicability of the concept because other factors may be found that contribute to the meaning of the concept as well. Factors can both plead in favour of, or against, the applicability of a concept. They differ from definitions and deeming provisions in that the conclusions based on factors can be (gradually) strengthened or weakened by conclusions of other factors. In contrast, conclusions based on definitions and deeming provisions cannot be strengthened or weakened by other factors; they are either accepted or rejected. In our specification of the DUBA only one concept is described with factors, namely the concept suitable employment (other vague notions in the DUBA are not addressed here). Note, that the concept of suitable employment is described with deeming provisions as well as factors.

4.2. Identification of Primitive Entities: Predicate Relations

So far in the creation of our expertise model we have performed two analyses. First, we identified the knowledge that had to be modelled in the domain specification, and in particular, what knowledge had to be recognised as domain knowledge and what had to be recognised as control knowledge. Second, we identified the structured entities of the domain specification, being the norms, events, processes, and concept descriptions. In this subsection we analyse the DUBA again for its concepts, but now with a less specific focus. We do not restrict ourselves to the concepts that have associated concept descriptions, but more generally to the concepts (and relations between concepts) that are used in the statute and in the world the statute assumes. For instance, the statute assumes there to be agreements between persons performing labour and persons and institutions who pay other persons to do so. We here denote such concepts with the term primitive entity as opposed to the structured entities described in the previous subsection (viz. norms, events, processes, and concept descriptions). In particular, we determine what in the domain should be seen as primitive entities, what attributes the primitive entities should be given, and what values the attributes can take. This analysis is an important step in choosing adequate predicate relations which in turn form an essential part of the domain ontology.

The analysis required to choose adequate predicate relations comprises two steps of the KANT method (Bench-Capon and Coenen, 1992): (a) the creation of two Freestyle KANT structures and (b) the creation of a class hierarchy on the basis of these Freestyle KANT structures.

(a) *Freestyle* KANT *structures*

There are two Freestyle KANT structures: the Tests-On-Objects structure (or: TOO structure) and the Entity-Attribute-Value structure (or: EAV structure). The TOO structure requires the DUBA to be analysed for the primitive entities and the tests

applied to them. For instance, if the statute refers to the employer of an employee this is interpreted as the test 'what is the employer of entity employee' (which we specify here as: 'employee has employer'). We applying KANT we use the following heuristics: (1) an entry in the TOO structure consists of a test carried out on an entity, (2) the test carried out on the entity is identified as an attribute to this entity, and (3) the result of the test is to be become a value of the attribute in the EAV structure. Below, we provide examples of the TOO structure created by applying the above mentioned heuristics:

- natural person has age
- employee has age
- employee has employer
- employer has delegate
- employer is located in
- labour relationship has wage
- labour relationship has employee
- labour relationship has terminated because
- labour offer has promotion prospect
- entitlement is based on statute
- benefit has volume
- reason is culpable

It is not always clear how an entry in the structure has to be chosen. We experienced two indeterminacies with respect to the creation of the TOO structure. The indeterminacies relate to (1) the choice to what entity an attribute (test) should be assigned, and (2) the choice between specifying a text fragment as an attribute or as a value of an (other) attribute. Below, we illustrate these indeterminacies.

The first indeterminacy relates to the choice to what entity an attribute has to be assigned. Consider the following example. According to De Wildt (1993), a job offer becomes less suitable for an employee whenever the acceptance of the offer implies taking the risk of losing the employees normal profession. Assume we have distinguished the entities 'employee' and 'labour offer'. The question is whether the test concerning the risk belongs to the entity 'employee' or to the entity 'labour offer'. In other words, should the 'risk of losing one's profession' be specified as an attribute of the entity 'employee' (attribute: 'the acceptance of a certain job offer implies taking a risk'), or as an attribute to the entity 'labour offer' (attribute: 'whenever a certain employee accepts this job, he or she is taking a risk'), or perhaps to both (which makes the structure redundant and gives a potential for inconsistency). There seem to be no objective criterion how to make a choice in such cases unless we consider task related issues, which we are concerned to avoid at this stage. In our analysis we therefore make a choice based on what seems more natural to us (in the example, we assign the attribute to the entity 'employee'). It

would be possible to guide such choices by adopting a suitable convention, such as assigning the attribute to the first of the entities mentioned in the legal text.

The second indeterminacy relates to the choice whether a concept should be specified as an attribute or as a value (cf. Bench-Capon et al. 1987). Although the TOO structure only refers to entities and attributes, the creation of the structure requires anticipation of the possible values of the attributes (which have to be specified in the EAV structure). Consider the following two entries in a TOO structure, showing alternative ways of assigning attributes to the entity 'labour':

1. labour is performed for private purposes,
   labour is performed for the purpose of a labour relationship,
   labour is performed for the purpose of an association or club,
2. labour is performed for the purpose of {p-1, ... p-n}.

In the EAV structure (created on the basis of the TOO structure) the three attributes of alternative (1) all may have values true and false, the test of alternative (2) may have more general values, such as: private life, labour relationship, or association/club. KANT does not provide guidelines as to what alternative is preferred here. Concerning the example, we have chosen the first alternative. In our opinion this more closely resembles the formulations in the texts because each of these tests occur separately at different locations in the text.* If the mutual exclusivity of the various purposes had been felt important, however, then the second version would have been preferred.

In summary, the indeterminacies must be dealt with by making pragmatic decisions (which are, however, recorded in the TOO structure so that they are open to inspection and modification rather that having to be reconstructed from the executable code). The resulting structure, the TOO structure, is the basis for the EAV structure. To create the EAV structure it is determined what values the attributes distinguished in the TOO structure can have. This determination requires decisions to be made about the range of dynamic aspects. For instance, we need to decide upon whether an employee can have more than one employer, and whether an employee can have more than one labour relationship with one employer (we assume both situations to be possible in our domain specification).

Some care has to be taken here. If we would specify the following EAV-structure entry (attributes which have their values marked with an asterisk may have multiple instantiations):

Entity	Attribute	Value
employee	has employers	{employer-1,...,employer-n}*

then, on the basis of this information, we are not able to determine which employer belongs to which relationship. For instance, if an employee emp has two labour

* Note that the alternative chosen depends on the way the text is stated. For instance, when in a text fragment an enumeration is given of labour-relation purposes, the second alternative would be more appropriate. Problems arise if both text fragments (1) and (2) can be found in the texts.

relations relation-1, and relation-2, with employers company-1 and company-2, respectively, then using the above mentioned EAV-structure entry we would specify:

Entity	Attribute	Value
emp	has employers	{company-1, company-2}

From this entry we cannot derive which employer belongs to which labour relation. To be able to relate an employees employer with labour relations we specify an 'intermediate' entity (which is a standard technique of data analysis in database systems for dealing with many-to-many relations). With an employer we also specify an identifier of the labour relationship. Henceforth, we specify the possible values of an attribute as a set of tuples (denoted with ⟨...⟩, for instance: {⟨relation-1, company 1⟩, ..., ⟨relation n, company n⟩}). Some entries of the EAV structure are given below.

Entity	Attribute	Values
employee	has labour offer	{⟨offer-1⟩,...,⟨offer-n⟩}*
employee	has age	{⟨age-1⟩,...,⟨age-n⟩}
employee	has employers	{⟨rel-1, employer-1⟩, ...⟨rel-n, employer-n⟩}*
labour	has volume	{⟨days-1, hours-1⟩, ...⟨days-n, hours-n⟩}
labour offer	has promotion prospect	{⟨good⟩, ⟨average⟩, ⟨bad⟩}
labour relation	has employer	{⟨employer 1⟩,...,⟨employer n⟩}
natural person	is employer	{⟨rel-1⟩,...,⟨rel-n⟩}*

(b) MIR *class hierarchy*

The second step in the KANT analysis is the creation of a MIR class hierarchy.* This hierarchy is obtained by ordering the entities in the EAV structure hierarchically, making abstractions and passing attributes that are shared by different entities up to their highest common level. The step turned out to encompass an extensive reinterpretation of the Freestyle structures and the DUBA. The reason is that the EAV structure contains redundant information. Below, we illustrate that the EAV structure is redundant by showing that the same information is stored with two different entities.

In several articles of the DUBA references are made to an employee. In article 3 for instance, a reference is made to the 'employer' of an 'employee', and in article 16 a reference is made to the 'labour' performed by an 'employee'. The EAV structure therefore contains the following two entries for the entity 'employee' (employers, labour, and labour relations are identified with constants employer-i, lab-i, rel-i, respectively):

employee	has employers	{⟨rel-1, employer-1⟩, ...,⟨rel-n, employer-n⟩}*
employee	performs labour	{⟨rel-1, lab-1⟩,...,⟨rel-n, lab-n⟩}*

* MIR is an abbreviation of MAKE INTERMEDIATE REPRESENTATION. MAKE, which stands for Maintenance Assistance for Knowledge Engineers, was a project carried out at the University of Liverpool (U.K.).

Beside these entries, the EAV structure also contains entries for the entity 'labour relation'. In the DUBA references are made to the 'employer' of a 'labour relation' (article 9), to the 'employee' of a 'labour relation' (article 4), and to the 'labour' that is performed in a 'labour relation' (article 4). Accordingly, the entity 'labour relation' has the following three entries in the EAV structure (each relation has only one employer, one employee and one labour identification):

```
labour relation has employer      {⟨employer-1⟩,...,⟨employer-n⟩}
labour relation has employee      {⟨employee-1⟩,...,⟨employee-n⟩}
labour relation labour            {⟨lab-1⟩,...,⟨lab-n⟩}
```

In these three entries, the same information is specified as in the entries for the entity employee as given above. Because the information is specified twice the EAV structure is redundant. In the creation of the MIR class hierarchy these redundancies have to be removed.

After removing redundant attributes and relocating some other attributes, the analysis revealed the following classes and subclasses: (1) natural persons, (1.1) natural-person employers, (1.2) employees, (2) institutions, (2.1) institution employers, (3) sites, (4) labour, (5) labour offers/relations, (5.1) labour offers, (5.2) labour relations (6) unemployment, (7) entitlements, (8) benefits, (9) reasons, and (10) statutes. Below, we list a fragment of the MIR class hierarchy:

```
Class 1:     natural persons
Attr/val:    age {0,...,150}
Attr/val:    name string

   Subclass 1.1:   employees
   Attr/val:       registered {⟨institution1⟩,...,⟨institutionn⟩}*
   Attr/val:       labour offers {⟨offer1⟩,...,⟨offern⟩}*

   Subclass 1.2:   natural-person employers
   Attr/val:       delegate {⟨del1⟩,...,⟨deln⟩}*

...

Class 5:     labour relations
Attr/val:    employer {⟨employer1⟩,...,⟨employern⟩}
Attr/val:    employee {⟨employee1⟩,...,⟨employeen⟩}
Attr/val:    labour {⟨labour1⟩,...,⟨labourn⟩}
Attr/val:    volume {0,...,100}

...

Class 8:     benefits
Attr/val:    statute {⟨unemployment-benefits-act⟩,
               ⟨act-on-minimum-wage⟩,...}
Attr/val:    volume {0,...,100}
Attr/val:    amount {0,...,10.000}
Attr/val:    begin date {⟨d1, m1, y1⟩,...,⟨dn, mn, yn⟩}
Attr/val:    end date {⟨d1, m1, y1⟩,...,⟨dn, mn, yn⟩}
```

4.3. Deriving Predicate Relations from the MIR Hierarchy

The MIR class hierarchy is an important step toward choosing predicate relations. The predicate relations can be derived from the hierarchy (we refer to the entries in the hierarchy as class-attribute-value entries) by applying the following three transformation heuristics (cf. McDermott, 1993, p. 56):

1. Class-attribute-value entries that deal with time are ignored. It is assumed that a set of predicates reflect one point of time.
2. Class-attribute-value entries of the form *class – attribute* – {⟨i-1-1,. . . ,i-1-n⟩, . . . ,⟨i-m-l,. . . ,i-m-n⟩} are specified as predicates: *name*(*Entity*, I-1,. . . ,I-n). Here, *name* is a unique name of the class-attribute-value entry, *Entity* is a variable that identifies an object in *class* (or a parent class of *class*), and I-1,. . . ,I-n are variables identifying values.
3. Asterisks in the hierarchy are ignored. Class-attribute-value entries that are marked with an asterisk (those that can have more than one value, such as an employee that is registered by more than one institution) will be specified by multiple clauses of the same predicate (e.g., *registered*(p, institution-1) and *registered*(p, institution-2)).

As an example of the application of the second heuristic* we mention the class-attribute-value entry *employee – registered* {⟨institution-1⟩,. . . ,⟨institution-n⟩} which is specified with predicate relation:

```
registered(NaturalPerson, Institution)
```

Sometimes, for convenience, different attributes are united into one single predicate, for instance the minimum and maximum wage according to a statute, or, the employer, the employee and the type of labour performed in a specific labour relation. The latter information is specified with the following predicate relation:

```
labour_relation(Relation, Labour, Employee, Employer)
```

The set of predicate relations obtained with the KANT analysis is assumed to be such that all relevant entities, attributes and values in the DUBA can be represented. In the next section we illustrate how the predicate relations derived are used for the domain ontology.

5. The Domain Ontology and Its Instantiation

The analysis so far has achieved the following things:

- we have identified the knowledge in the DUBA we wish to represent;
- we have separated control knowledge and domain knowledge (as defined in the CommonKADS expertise model);

* We deem the application of heuristic 1 and 3 to be self-evident.

- we have identified the particular norms, acts and concept descriptions that must be represented. Further, since these are to be represented as frame structures as defined by Van Kralingen, we have a complete set of slots for which information is required;
- we have defined a vocabulary which we can use to fill the slots.

In principle, this completes the ontology with which the domain knowledge can be expressed. We note that the ontology consists of a generic part (viz. the frame structures which can be used throughout the legal domain) and a statute-specific part (viz. the vocabulary of the DUBA determined with KANT). So far, the ontology – and in particular the generic part of the ontology – have only been specified in an informal language. Before we express the knowledge from the DUBA in COMMONKADS domain models we formalise both the statute specific and the generic ontology. In this process, the frame structures are modified to some extent (viz. some slots are added and some get a slightly different interpretation to support automated reasoning). For reasons of brevity we do not elaborate on this formalisation process. A fuller account is given in Visser (1995a) and Visser and Bench-Capon (1996). We here assume there to be formal frame structures and formal languages to fill the frame structures.

The slots in these frames are intended to record all the attributes possessed by the entities they represent. Various tasks may use information contained in different slots but the frame is intended to record all the information which may be required by any task. Below, we provide examples of frames contained in the domain models. The scope of this paper does not allow us to explain the frames in full detail.

Norm frame: The norm frame below specifies that an employee ought not become culpably unemployed (Dutch Unemployment Benefits Act, art. 24, sec. 1a). It is considered a norm of type conduct.

```
norm identifier:      n_DUBA_24_1a
norm type:            conduct
promulgation:         {{date(1986,11,6), source(statute), id(DUBA),
                        art(24), sec(1), sub(a)}}
scope:                {source(statute), id(DUBA)}
time reference:       T
object conditions:    true_from(T, employee(P,R)) and
                        true_from(T, labour_relationship(R,A,P,G))
meta conditions:      always_true
subject:              P
legal modality:       ought_not
act reference:        le_become_culpably_unemployed(P,R)
```

In this definition R is an identifier for the labour relationship concerned, A is an identifier for the labour performed in the labour relationship, and G is an identifier for the employer of employee P (note the addition of a time-reference slot, which is used for temporal reasoning).

Act frame: Two types of act frames are distinguished; event frames and process frames. Whereas the former type of frame specifies acts that occur instantaneously, the latter type of frame specifies acts that are considered to have a duration. We give an example of an event frame that specifies the (physical) event of submitting a request for a benefit (Dutch Unemployment Benefits Act, art. 26, sec. 1b):

```
event identifier:       pe_submit_request
act:                    pe_submit_request(P,E)
promulgation:           {{date(1986,11,6), source(act), art(26),
                          sec(1), sub(b)}}
scope:                  {source(statute), id(DUBA)}
agent:                  P
act type:               physical
temporal setting:       true_from(Tb, unemployed(P,E,H)) and
                          function(days_after(E,8,Tmx))
                          and between(E,Te,Tmx)
spatial setting:
circumstantial setting:
time reference:         [Tb, Te]
initial state:          not(request_for_benefit(P,E))
final state:            request_for_benefit(P,E)
```

In this definition E is an identifier for the benefit (as well as an identifier for the first day of unemployment). H is the number of hours for which the employee is considered unemployed, variables beginning with T denote various times.

Concept frame: The concept frame given below specifies that whenever someone has a labour relationship in the military service, then the Minister of Defence is his employer (Dutch Unemployment Benefits Act, art. 4, sec. 1g, and art. 10, sec. a).

```
concept identifier:   DUBA_4_1g_DUBA_10a
concept:              employer(G,R)
concept type:         definition
priority:
time reference:       T
promulgation:         {{date(1986,11,6), source(art), id(DUBA),
                        art(4), sec(1), sub(g)}
                        {date(1986,11,6), source(art), id(DUBA),
                        art(10), sec(a)}}
scope:                {source(statute), id(DUBA)}
conditions:           true_from(T, labour_relation(R,A,P,G)) and
                        true_from(T, military_service(A)) and
                        true_from(T, minister_of_defence(G))
instances:            always_false
```

In this definition R is an identifier for the labour relationship, E is the employer of employee P, and A is an identifier for the labour performed in labour relationship R.

The formal frame structures have been used in the construction of a prototype legal knowledge system, called FRAMER. The translation of the formal frame structures into a knowledge representation (using prolog as a representation language)

is straightforward and can be done automatically. FRAMER is capable of performing both a planning task and an assessment task using the same domain specification. For details on FRAMER we refer to Visser (1995b).

6. Concluding Remarks

In this paper we have presented a method for conceptualising legal domains, which can be used in the construction of legal knowledge-based systems. The method provides an ontological framework to create intermediate specifications of legal domain knowledge (the expertise models). The method has been applied to create an expertise model of a substantial fragment of the Dutch Unemployment Benefits Act. Thereafter, we used the expertise model in the creation of a prototype system (FRAMER), which performs the tasks of assessment and planning in the domain (see Visser, 1995a, chapters 5 and 6; 1995b).

Our aim in presenting this material is to provide an example of the sort of implementation independent model that can serve as the foundation for an LKBS. A properly-engineered knowledge-based system requires the detailed analysis and design just as a properly-engineered conventional system does. In the early days of knowledge-based systems this was not realised: it was thought that recording the knowledge of the expert in a declarative representation would enable this kind of design and analysis to be avoided. But recording the knowledge is not enough, it must also be welded into a coherent whole which brings us back to the need for the detailed analysis and design. There are no easy routes to building a robust and well-engineered system. The design and analysis is needed:

- to provide an *ontological framework in which the knowledge can be assimilated and brought together.* In a system the various items of knowledge must work together, and be compatible with one another. A monolithic collection of rules does not satisfy this requirement: adding a new rule offers no guarantee that it will work well with the others.
- to provide *guidelines as to what knowledge must be acquired.* A piece of legislation is not free standing, but is to be read and interpreted against a background understanding of the nature of law, and of the language in which the law is written. No such background is available to a knowledge-based system, and so the additions necessary to make it a free standing representation must be identified and supplied. Our method identifies where the additional information is needed: potential gaps with respect to law are identified by the existence of the slots of the frames which have not been supplied with fillers, and assumption masked by the use of natural language are uncovered during the application of the KANT methodology.
- to provide a means of *recording and presenting the various design choices* that are made during the modelling process. Even if a methodology were to eliminate the necessity for making choices – which it cannot – it would still be necessary to record the choices that are dictated by the methodology. A

record of these choices is essential for the system verification, validation and maintenance, and if the model is to be reused across a variety of applications.

- to help *eliminate task bias* in the acquisition process. A major strength of our approach is a clear separation between knowledge of the domain and knowledge of how that domain knowledge can be manipulated to solve problems in the domain (viz. the control knowledge). For this to be successful it is important that what we believe to be domain knowledge really is domain knowledge and is not skewed by a problem-solving perspective.

To date there have been few such analysis and design studies carried out in a thorough way as a precursor to constructing a knowledge-based system in the legal domain. There have also been very few successful knowledge-based systems in the legal domain. We believe that there may be a connection between these two things.

Finally, we should make some remarks as to how generally applicable our method is. We believe that the basic division of the domain into acts, norms and concepts is of entirely general application. This forms the basis of a generic ontology for legal domains: while we do not claim that this is the only way in which legal domains can be conceptualised, we do believe that it can form the basis of a conceptualisation of any legal domain. Also the method can be applied to any legal domain, although as it goes into greater detail the ontology it yields will become specific to the particular area of law being analysed. As a *method*, however, it is of general applicability.

Acknowledgements

The research presented in this paper has partly been carried out at the Department of Law and Computer Science of the University of Leiden (The Netherlands). The study at the University of Leiden was supported by a grant from the Foundation for Law and Public Administration (REOB) which is part of the Netherlands Organization for Scientific Research (NWO). The authors wish to thank Robert van Kralingen for his contribution to the ideas presented in this paper.

References

Aamodt, A., Bredeweg, B. Breuker, J.A., Duursma, C. Löckenhoff, C., Orsvarn, K., Top, J., Valente, A. & W. van de Velde, W. (1992). *The CommonKADS Library*. Document KADS-II/T1.3/VUB/TR/005/1.0, Amsterdam.

Aarnio, A. (1987). *The Rational as Reasonable; A Treatise on Legal Justification*. Law and Philosophy Library. D. Reidel Publishing Company: Dordrecht.

Allen, J.F. (1984). Towards a General Theory of Action and Time. *Artificial Intelligence* **23**(2): 123–154.

Andringa, L., Bandringa, J. & Vos, P. (1987). *De Werkloosheidsuitkering na de Stelselherziening*, Sociale Verzekeringswetten. Kluwer: Deventer [in Dutch].

Ashley, K.D. (1990). *Modelling Legal Argument*. MIT Press: Cambridge, MA.

Bench-Capon, T.J.M. (1987). Support for Policy Makers: Formulating Legislation with the Aid of Logical Models. In Proceedings of *The First International Conference on Artificial Intelligence and Law*, 181–189. Boston, MA.

Bench-Capon, T.J.M. & Coenen, F.P. (1992). Isomorphism and Legal Knowledge Based Systems. *Artificial Intelligence and Law* **1**(1): 65–86.

Bratley, P., Fremont, J., Mackaay, E. & Poulin, D. (1993). Coping with Change. In Proceedings of *The Third International Conference on AI and Law*, 69–76. ACM Press: New York.

Breuker, J.A. & den Haan, N. (1991). Separating World and Regulation Knowledge: Where Is the Logic? In Proceedings of *The Third International Conference on Artificial Intelligence and Law*, 92–97. Oxford.

Breuker, J.A. & van de Velde, V. (1994). *CommonKADS Library for Expertise Modelling, Reusable Problem Solving Components*. IOS Press: Amsterdam.

Brouwer, P.W. (1990). *Samenhang in recht, Een analytische studie*, Rechtswetenschappelijke reeks. Wolters-Noordhoff: Groningen [in Dutch].

Brownston, L., Farell, R., Kant, E. & Martin, N. (1985). *Programming Expert Systems in OPS5*. Addison Wesley: Reading, MA.

Bylander, T. & Chandrasekaran, B. (1987). Generic Tasks for Knowledge-Based Reasoning: The "Right" Level of Abstraction for Knowledge Acquisition. *International Journal of Man-Machine Studies* **26**: 231–243.

Clancey, W.J. (1986). Representing Control Knowledge as Abstract Tasks and Metarules. In Coombs, M.J. & Bolc, L. (eds.) *Computer Expert Systems*. Springer-Verlag: New York.

Coenen, F.P. & Bench-Capon, T.J.M. (1993). *Maintenance of Knowledge Based Systems: Theory, Techniques and Tools*. Academic Press: London.

Cohen, P.R. & Feigenbaum, E.A. (1983). Planning and Problem Solving. In Cohen, P.R. & Feigenbaum, E.A. (eds.) *The Handbook of Artificial Intelligence*, Vol. III, 516–562. Pitman Books: London.

Davis, R. (1980). Meta-Rules: Reasoning about Control. *Artificial Intelligence* **15**: 179–222.

Franken, H. (1987). *InLeiden tot de Rechtswetenschap*. Gouda Quint: Arnhem [in Dutch].

Franken, H. (1988). Systeemtheorie en rechtswetenschap. In Klanderman, J.H.M. & Roos, N.H.M. (eds.), *Interne en Externe Analyses van Recht*, 205–247. W.E.J. Tjeenk Willink: Zwolle [in Dutch].

Gardner, A. v.d. L. (1987). *An Artificial Intelligence Approach to Legal Reasoning*. The MIT Press: Cambridge, MA. Kingdom.

Georgeff, M.P. (1987). Planning. *Annual Review of Computer Science* **2**: 359–400; also in Allen, J., Hendler, J. & Tate, A. (eds.) *Readings in Planning*, 5–25. Morgan Kaufmann Publishers: San Mateo, CA.

Groendijk, C. & Oskamp, A. (1993). Case Recognition and Strategy Classification. In Proceedings of *The Fourth International Conference on Artificial Intelligence and Law*, 125–132.Amsterdam.

Gruber, T.R. (1992). *Ontolingua: A Mechanism to Support Portable Ontologies*. Technical report KSL 91-66, Stanford University, Knowledge Systems Laboratory, Stanford, U.S.A.

Hart, H.L.A. (1961). *The Concept of Law*, Clarendon Law Series. Oxford University Press. Oxford.

Hayes-Roth, B. (1988). A Blackboard Architecture for Control. In *Readings in Distributed Artificial Intelligence*, 505–540. Morgan Kaufmann Publishers: San Mateo, CA.

Heijst, G. van (1995). *The Role of Ontologies in Knowledge Engineering*. Doctoral Thesis, University of Amsterdam.

Kowalski, R.A. & Sergot, M.J. (1986). A Logic Based Calculus of Events. *New Generation Computing* **4**(1): 67–95; also in Thanos & Schmidt (eds.) *Knowledge Based Management Systems*. Springer-Verlag: Heidelberg.

Kralingen, R.W. van & Visser, P.R.S. (1991). Een strategisch interactief expertsysteem voor het opstellen van tenlasteleggingen. In van der Spek, B.R. & Treur, J. (eds.) *AI Toepassingen 1991*, 163–173. Vrije Universiteit Amsterdam: Amsterdam [in Dutch].

Kralingen, R.W. van (1995). *Frame-Based Conceptual Models of Statute Law*, Computer/Law Series, Vol. 16. Kluwer Law International: The Hague.

McDermott, D. (1993). Book Review: D.B. Lenat, and R.V. Guha, Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project. *Artificial Intelligence* **61**: 53–63.

Newell, A. (1982). The Knowledge Level. *Artificial Intelligence* **18**: 87–127.

Oskamp, A. (1993). Model for Knowledge and Legal Expert Systems. *Artificial Intelligence and Law* **1**(4): 245–276.

Prakken, H. (1993). *Logical Tools for Modelling Legal Argument*. Doctoral Thesis, Vrije Universiteit Amsterdam, Amsterdam.

Routen, T.W. & Bench-Capon, T.J.M. (1991). Hierarchical Formalizations. *International Journal of Man-Machine Studies* **35**: 69–93.

Ruiter, D.W.P. (1993). *Institutional Legal Facts; Legal Powers and their Effects*, Law and Philosophy Library. D. Reidel/Kluwer Academic Publishers: Dordrecht.

Sartor, G. (1992). Normative Conflicts in Legal Reasoning. *Artificial Intelligence and Law* **1**(2–3): 209–235.

Sergot, M.J., Cory, H.T., Hammond, P., Kowalski, R.A., Kriwaczek, F. & Sadri, F. (1986). Formalisation of the British Nationality Act. In Arnold, C. (ed.) *Yearbook of Law, Computers and Technology*, Vol. 2. Butterworths: London.

Tan, Y.H. & Treur, J. (1991). *Constructive Default Logic and the Control of Defeasible Reasoning*. Report IR-280, Vrije Universiteit Amsterdam, Amsterdam.

Tate, A., Hendler, J. & Drummond, M. (1990). A Review of AI Planning Techniques. In Alen, J., Hendler, J. & Tate, A. (eds.) *Readings in Planning*, 26–49. Morgan Kaufmann Publishers: San Mateo, CA.

Thornton, G.C. (1987). *Legislative Drafting*, 3rd edition. Butterworths: London.

Valente, A. (1995). *Legal Knowledge Engineering; A Modelling Approach*. Doctoral Thesis, University of Amsterdam, Amsterdam. IOS Press, Ohmsa: Amsterdam.

Vermesan, A.I. & Bench-Capon, T.J.M. (1995). Techniques for the Verification and Validation of Knowledge Based Systems: A Survey Based on the Symbol/Knowledge Level Distinction. *Software Testing, Verification and Reliability*, in press.

Visser, P.R.S. (1995a). *Knowledge Specification for Multiple Legal Tasks; A Case Study of the Interaction Problem in the Legal Domain*, Computer/Law Series, Vol. 17. Kluwer Law International: The Hague.

Visser, P.R.S. (1995b). *FRAMER: Source Code of a Legal Knowledge System That Performs Assessment and Planning*, Reports on Technical Research in Law, Vol. 2, No. 1. University of Leiden: Leiden.

Visser, P.R.S. & Bench-Capon, T.J.M. (1996). The Formal Specification of a Legal Ontology. In Proceedings of *The Ninth International Conference on Legal Knowledge-Based Systems (JURIX'96)*. Tilburg, to appear.

Wielinga, B.J., van de Velde, W., Schreiber, A.Th. & Akkermans, J.M. (1992). *The CommonKADS Framework for Knowledge Modelling*, deliverable KADS-II/T1.1/PP/UvA/35/1.0. University of Amsterdam: Amsterdam.

Wildt, J.H. de (1993). *Rechters en Vage Normen, een jurimetrisch onderzoek naar de uitleg van het begrip 'passende arbeid' uit de Werkloosheidswet*. Gouda Quint: Arnhem [in Dutch].