TR-0883

Constructing a Legal Knowledge-base with Partial Information

by
T. Nishioka (MRI), K. Yokota
C. Takahashi (JIPDEC) & S. Tojo (MRI)

July, 1994

Constructing a Legal Knowledge-base with Partial Information

Toshihiro Nishioka Kazumasa Yokota [*]
Chie Takahashi [†] Satoshi Tojo

Information Science Dept., Mitsubishi Research Institute Inc.
ARCO TOWER Bldg. 9F., 1-8-1, Shimomeguro, Meguro-ku, Tokyo 153, Japan
e-mail: {nishioka,tojo}@mri.co.jp

Keywords: knowledge representation, knowledge-base construction, partial information, conditional query and answer, deductive object-oriented database

Abstract

In legal reasoning systems, a typical application of normative reasoning, partial information plays an important role in the representation and reasoning of legal knowledge. To construct a legal knowledge-base with partial information, many features are required of knowledge representation languages. In this paper, we discuss the representation of knowledge-bases and their refinement through our experimental system, TRIAL. The system is based on the $\mathcal{QUIXOTE}$ deductive object-oriented database language. In particular, we point out that three features (knowledge modularization, hypothetical reasoning, and hypothesis generation (abductive reasoning)) are indispensable to the construction of a legal knowledge-base.

1 Introduction

Recently, legal reasoning, a typical example of normative reasoning, has attracted much attention in the field of artificial intelligence. Legal reasoning systems are applications, whose development, like that of theorem provers, dates back to before artificial intelligence was proposed, (see [11]). In fact, laws are closely related to the judicial world and all social activities. To support legal interpretation and reasoning in a wide range of situations, many systems have been developed, including those capable of planning tax-saving strategies, negotiating the payment of damages, making contract documents, predicting judgments, and supporting legislation. Although many works on expert systems for such applications have been published, a powerful legal database system has not yet been reported.

The Japanese FGCS (Fifth Generation Computer System) project considered legal reasoning systems quite critical and developed the TRIAL prototype legal reasoning system[16,

[*]Institute for New Generation Computer Technology (ICOT), e-mail: kyokota@icot.or.jp
[†]Japan Information Processing Development Center (JIPDEC), e-mail: j-takaha@icot.or.jp

17, 20, 22]. For the TRIAL system, we provided the *QUIXOTE* [21]. *QUIXOTE* is a deductive object-oriented database (DOOD) [5, 6] language which may be used for knowledge representation and management.

From our experience, we concluded that *partiality of information* plays an important role in legal reasoning systems. For example, in designing a new case knowledge-base, we were confronted with the following problems: the data structures of a new case cannot be specified in advance, attributes can have indefinite values, and the data itself can be ambiguous or inconsistent. That is, a new case model can have partial information: the details relating to various important points in the problem domain may be insufficient. Similar problems are encountered when constructing other legal knowledge-bases for information such as precedents.

The following example shows many of the problems we confronted in constructing a knowledge-base with partial information. Some features of *QUIXOTE* are useful for processing such partial information. TRIAL uses the *QUIXOTE* language to model data and knowledge as objects with partial information. Some advanced query processing mechanisms, such as hypothetical reasoning and hypothesis generation, are also used for refining a partial information knowledge-base.

In this paper, we report on the TRIAL system and how we construct a legal knowledge-base using the advanced knowledge processing features of a DOOD system. In Section 2, we present a legal reasoning example to illustrate the kind of problems we confronted. Section 3 provides a brief explanation of *QUIXOTE*. Section 4 explains the design strategy, contents, and a series of queries for an example knowledge-base.

2 An Example from Legal Reasoning

In this section, to concretely illustrate the types of problems we are studying, we examine the following new case related to "*karōshi*" (death from overwork):

> Mary, a driver employed by company 'S,' died from a heart attack while taking a break between jobs. Can this case be applied to the worker's compensation law?

We will first give a brief summary of the legal reasoning process we adopted. Next, we will introduce part of the legal knowledge related to this example, and the most appropriate interaction sequence between a user and a knowledge-base management system for dealing the example. And last, we will present some requirements for a knowledge representation language, as deduced from this example.

2.1 Legal Reasoning Process

We decompose the analytical legal reasoning process into three steps: *fact finding*, *statutory interpretation*, and *statutory application*. Although fact finding is very important, it is beyond the capabilities of currently available technologies. So, we assume that any new cases are already represented in a form that is compatible with our system. Statutory interpretation is a particularly interesting theme from an artificial intelligence point of view. TRIAL focuses on both statutory interpretation and statutory application.

Among the many approaches to statutory interpretation, we decided to apply the following procedure:

1. *Analogy detection*
   Given a new case, precedents having similarities to the case are retrieved from an existing precedent database.

2. *Rule abstraction*
   Precedents (interpretation rules), extracted by analogy detection, are abstracted until the new case can be applied to them.

3. *Deductive reasoning*
   The new case is applied in a deductive manner, to the abstract interpretation rules transformed in the previous step. This step may include statutory application because it is used in th. same manner.

Among these steps, the analogy detection strategy is essential to legal reasoning to enable the *more efficient* detection of *better* precedents. Analogy detection ultimately determines the quality of the result. As the primary objective of TRIAL is to investigate the capabilities of *QUIXOTE* in this area and develop a prototype system, we have limited the scope of our present study. That is, we have chosen to investigate the extent to which interpretation rules should be abstracted for *a* new case, to obtain plausible answers. We have not attempted to devise a general abstraction mechanism.

2.2 Example

In this example, we use a statute and a theory for its application:

- *labor law*: An organization is responsible for employee compensation, if the case judgment is for 'insurance.'
- *theory*: If the case judgment is for both 'job-causality' and 'job-execution', then the case judgment is for 'insurance.'

Assume that there are two precedents related to the law which have already been abstracted as follows:

- *precedent 1 (job-execution)*: If an *employee* has a *relation* of employment, and this relationship causes the *case*, then the judgment considers the case as being part of 'job-execution.'
- *precedent 2 (job-causality)*: In the *case*, if a *disease*-related incident occurred within the *job*'s period, then the judgment considers the case to be 'job-causal.'

Note that these statements are abstracted from certain concrete precedents in the rule abstraction step by abstracting concrete concepts (e.g., a case name, or a person's name) into abstract concepts (shown in italics): *employee, relation, case, disease*, and *job*. We will introduce our implementation of rule abstraction in section 4.

Finally, for the above knowledge-base, we consider queries and expected answers.

- *query 1*: According to past precedents and theory, what kind of judgment can we predict for the new case?
- *answer 1*: If, in the new case, Mary's activities are work-related and they are the cause of the new case, the judgment is for 'insurance.'
- *query 2*: According to labor law, what responsibility does Mary's company have?
- *answer 2*: If, in the new case, Mary's activities are work-related and they are the cause of the new case, company 'S' is responsible for compensation.

2.3 Requirements of Knowledge Representation Languages

To realize the above system, we consider the following advanced features:

A. *Classification of knowledge*: Situation-dependent knowledge and inconsistent knowledge should be managed.

B. *Supply of any lacking information*: Hypotheses may be needed to supply lacking information. For example, a break between jobs might be considered as being part of the job.

C. *Inference of lacking information*: The knowledge-base management system must generate any information which is lacking. For example, in the previous section, the clause: "If in the new case, Mary's activities are work-related and they are the cause of the new case" is a generated hypothesis.

D. *Selection of knowledge resources*: The knowledge-base management system has to provide a function capable of selecting knowledge modules that are appropriate for the query. For example, the part "according to $\sim$" in the query in section 2.2 means that they are directed toward specific knowledge modules.

E. *Trial and error environment*: If features B, C, and D, above, are provided, users may want to scrap existing knowledge modules and build their own temporary modules to test hypotheses or classified knowledge configurations. Using such new modules, users can issue queries to examine the plausibility and legality of their legal contentions.

F. *Versioning knowledge*: Users may want to impose version control on the knowledge-base to enable its incremental construction.

G. *Abstracting/specializing rules and knowledge modules*: In section 2.2, abstracted rules must be developed from the concrete rules in the precedent database.

H. *Analogical matching of concepts*: In analogy detection, users may want to compare concepts for analogies.

In next section, we will explain how *QUIXOTE* can (or cannot) realize the above requirements for constructing and managing knowledge-bases.

3 Overview of *QUIXOTE*

From a database point of view, *QUIXOTE* is a DOOD language, while, from a logic programming point of view, it is an extended constraint logic programming language based on subsumption constraints. In this section, we explain some of its features, used in the above example. For details of *QUIXOTE*, see[18, 21, 22].

3.1 Object Identity and Subsumption Relation

Concepts are represented by *object terms* in *QUIXOTE* [1]. *mary, driver* and *employee* are simple examples of object terms, while *company* [*name*= "*S*"] is a slightly more complex example, representing "a company whose name is 'S'."

Object terms are partially ordered by the *subsumption relation* '$\sqsubseteq$'. For example,

$$mary \sqsubseteq driver, \quad driver \sqsubseteq employee, \quad company\,[name{=}\text{"}S\text{"}\,] \sqsubseteq company.$$

Relations between concepts such as "Mary is a driver," and "heart attack is a kind of disease" can be represented by this partial order.

3.2 Subsumption Constraints

To represent relations such as "Mary is employed by company 'S' ", we use a *subsumption constraint*:

$$mary.employer \cong company\,[name{=}\text{"}S\text{"}\,],$$

where *mary.employer* is called a *dotted term* and represents 'Mary's employer', and $A \cong B$ means that $A \sqsubseteq B$ and $B \sqsubseteq A$.

We use an abbreviation $o/[l \rightarrow t]$ to represent $o \mid \{o.l \sqsubseteq t\}$, as well as $o/[l \leftarrow t]$ for $o \mid \{o.l \sqsupseteq t\}$ and $o/[l = t]$ for $o \mid \{o.l \cong t\}$. These are called *attribute terms*.

3.3 Rules

A rule is defined as follows:

$$a_0 \Leftarrow a_1, \cdots, a_n \parallel D,$$

where $a_0, a_1, \cdots, a_n$ are attribute terms and D is a set of subsumption constraints. a_0 is called a *head*, $a_1, \cdots, a_n \parallel D$ is called a *body*, and a_i's are called *subgoals*. A rule means that if the body is satisfied, the head is satisfied. For example,

$$\begin{aligned}
&compatriots\,[person_1 = X,\ person_2 = Y] \\
&\qquad \Leftarrow X/[nationality{=}\ N_1], Y/[nationality{=}\ N_2] \\
&\qquad\qquad \parallel \{X \sqsubseteq person,\ Y \sqsubseteq person,\ N_1 \sqsubseteq nation,\ N_2 \sqsubseteq nation,\ N_1 \cong N_2\};;
\end{aligned}$$

means that if two persons X and Y have a same nation as their nationalities, there is a relation '*compatriots*' between them. A rule without a body is called a *fact*.

[1]Although *QUIXOTE* can handle object sets as terms, we do not describe this here.

3.4 Realization of the Requirements

We now show how the requirements for the knowledge-base system, proposed in section 2.3, are satisfied (or otherwise) by $\mathcal{QUIXOTE}$.

$\mathcal{QUIXOTE}$ supports three powerful features for handling partial information. Let us consider their utilities.

1. *Knowledge modularization*:

   The concept of knowledge modularization is important in the field of knowledge representation [13]. $\mathcal{QUIXOTE}$ allows sets of rules to be modularized:

   $$m :: \{r_1, \cdots, r_n\},$$

   where m is an object term called a *module identifier* (mid) and $r_1, \cdots, r_n$ are rules.

   The definition of rules is extended to the external reference of objects:

   $$a_0 \Leftarrow m_1 : a_1, \cdots, m_n : a_n \parallel D,$$

   where $m_1, \cdots, m_n$ are module identifiers. This rule means that if a_i and D are satisfied in module m_i for $1 \leq i \leq n$, then a_0 is also satisfied. If module m contains the rule and a_0 $(= o_0 \mid C_0)$ is satisfied, we use the expression: "o_0 *exists* in m and has C_0 in m."

   Rules are imported and exported by *rule inheritance*, defined in terms of the binary relation (written as '$\sqsupseteq_S$') between modules, called a *submodule relation*. If $m_1 \sqsupseteq_S m_2$, m_1 *inherits* all rules in m_2. The right hand side of '$\sqsupseteq_S$' can be a formula consisting of module identifiers and set operators: '$\cup$' (union) and '$\setminus$' (difference).

   This feature is strongly related to the requirements A, D, and F:

   A. *Classification of knowledge*
      This actually requires static modularization of the knowledge, and can be realized by feature 1.

   F. *Versioning knowledge*
      This can be partly realized by the same feature. A module in $\mathcal{QUIXOTE}$ can be used to represent a version of the knowledge. This is done by adopting some conventions to represent a version in a module identifier, e.g. all module identifiers must be like *labor-law* [*version*=2]. When we say 'partly realized', we mean that $\mathcal{QUIXOTE}$ does not provide any of the special version control functions already provided by widely used version control systems.

   D. *Selection of knowledge resources*
      Using feature 1, a user can select knowledge resources by selecting knowledge modules.

   In [9], an ATMS-based expert system is described as it deals with inconsistency by controlling inconsistent knowledge modules, each of which is itself consistent. Actually, $\mathcal{QUIXOTE}$ provides a similar facility for handling inconsistencies. For example, several precedents which have (apparently) contradicting legal contention can be represented by providing a knowledge module for each precedent. Furthermore, *QUIXOTE* can handle local inconsistencies within a knowledge module by allowing objects to have properties whose values are $\bot$ (inconsistent) and enabling the computation mechanism of *QUIXOTE* to deal with $\bot$.

2. *Hypothetical reasoning:*

   Query processing in *QUIXOTE* corresponds to resolution and constraint solving in constraint logic programming.

   A *query* is defined as a pair (A, P) (written ?-A; ; P) of a set A of attribute terms and a program P, where A is referred to as the *goal* and P as the *hypothesis*. If the query is issued to a database DB, the meaning is 'if P is in DB, is A?'. Although that sounds like a subjunctive query, like that below, *QUIXOTE* treats it in a simpler way than other current research into this topic [7].

   A *database* or a *program* is defined as the triplet (S, M, R) of a finite set S of subsumption relations, a set M of submodule relations, and a set R of rules.

   Consider a database DB. A query ?-A; ; P to DB is equivalent to query ?-A to $DB \cup P$ (If $DB = (S_1, M_1, R_1)$ and $P = (S_2, M_2, R_2)$ then $DB \cup P = (S_1 \cup S_2, M_1 \cup M_2, R_1 \cup R_2)$). That is, P is inserted into DB before A is processed. In other words, P works as a hypothesis for ?-A.

   As hypotheses are incrementally inserted into a database, nested transactions are introduced to control such insertions. See [21] for details.

   This feature is related to the requirements B, D, E, and G:

   B. *Supply of any lacking information*
   This is satisfied by feature 2 by definition.

   D. *Selection of knowledge resources*
   To satisfy this requirement, a little more functionality is required than that provided by feature 1. That is, without the dynamic configurability of knowledge modules, a user cannot freely merge modules or issue queries relating to them. Feature 2 supports this function, because hypotheses permit new module definition and addition of submodule relations.

   E. *Trial and error environment*
   More functionality is needed to satisfy this requirement: it must be easier to construct a knowledge-base using a 'trial and error' style. As mentioned in the explanation of feature 2, *QUIXOTE* provides nested transactions which allow users to easily roll back any changes to the knowledge-base within the transactions. Furthermore, as rules can be marked as not being inherited by submodules, and/or inherited rules can be overridden if they have the same object terms in their head, various knowledge configurations can be examined.

   G. *Abstracting/specializing rules and knowledge modules*
   If a rule can be parameterized, rule abstractions and specializations become possible by adding a parameter module as the hypothesis. As shown in section 4, TRIAL adopts this method.

3. *Hypothesis generation:*

An *answer* of *QUIXOTE* is defined as the triplet (D, V, E). D, which is called the assumption part of the answer, is a set of subsumption constraints that cannot be solved during query processing; V, the conclusion part of the answer, is a set of variable constraints that are bound during query processing; and E, the explanation part of the answer, is the corresponding derivation flow. Note that only subsumption constraints of object properties can exist in the assumption part.

This feature is related to requirement C:

C. *Inference of lacking information*
This is satisfied by feature 3 by definition. The assumption part given by this feature is actually the information which is lacking and obtained by abductive reasoning.

HYPO [2], a kind of case-based reasoning system, has a similar facility for retrieving important relevant information through abductive reasoning. A *dimension* in HYPO represents a relationship between a cluster of facts and a legal consequent based on those facts. Dimensions for which all or part of the antecedents are satisfied are used in inferences. A dimension where not all antecedents are satisfied is called a near-miss dimension, and is processed by abductive reasoning. Although HYPO and *QUIXOTE* are quite similar in their abductive function, there are some differences in the design of the abduction strategy. While HYPO is designed especially to produce legal arguments, *QUIXOTE* is a general purpose knowledge-base facility, such that *QUIXOTE*'s abduction function can simulate that of HYPO.

One remaining requirement, H. *Analogical matching of concepts*, is a very difficult problem which has not been realized in *QUIXOTE*. This function is implemented in the TRIAL system by providing hints of an analogy between concepts with the subsumption hierarchy of basic objects.

4 Legal Reasoning in *QUIXOTE*

In this section, we explain the TRIAL legal reasoning system which we implemented in *QUIXOTE*. By showing the overall architecture, a design strategy of a knowledge-base, and an implementation of the example knowledge-base on TRIAL, we demonstrate *QUIXOTE*'s ability to construct knowledge-bases with partial information.

4.1 Implementation of TRIAL

The overall system architecture is shown in Figure 1.

Note that all the data and knowledge in the database component is written in *QUIXOTE*.

4.2 A Strategy for Designing a Knowledge-base in *QUIXOTE*

In this section, we present the strategy adopted in the design of our example knowledge-base: the construction of an initial model, and its subsequent refinement. These methods are neither formal nor scalable to large-scale knowledge-bases, but provide a practical

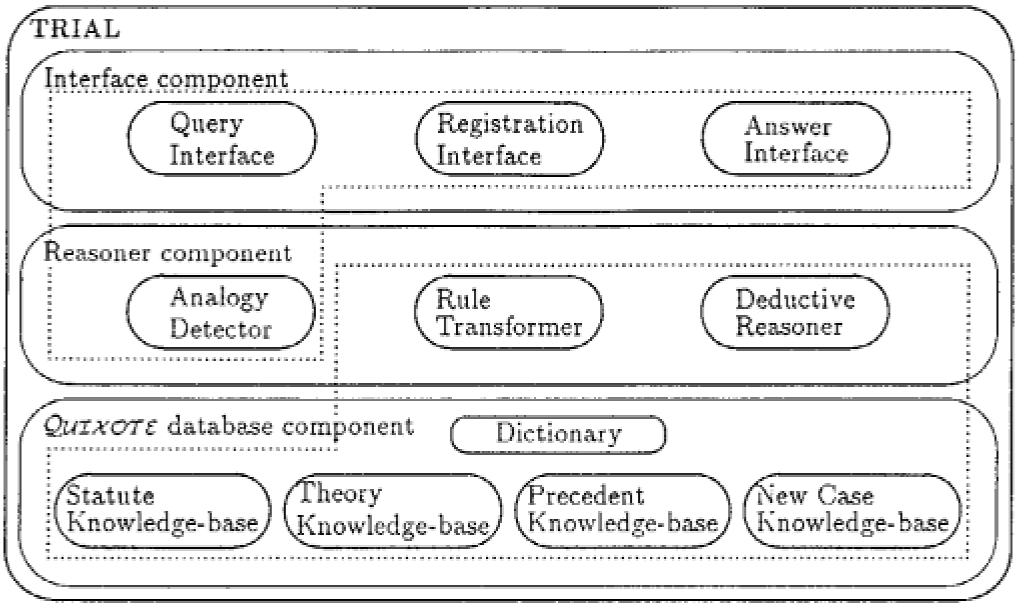

Figure 1: Architecture of TRIAL

means of solving limited problems related to constructing (legal) knowledge-bases. These methods should be extended and scaled-up to enable their application to practical situations.

Creating an Initial Model

To create an object-oriented model, we must specify

- what objects are, and
- the relation that exists between objects

in the earliest stage of the modeling [10]. Although there are various modeling methodologies [14, 15], most of this important work is based on experience and intuition. Since 𝒬𝓊𝒾𝓍𝑜𝓉ℯ supports a mechanism enabling hypothesis generation, however, the development of a strategy for hypothesis generation may be beneficial to the overall design.

As shown in figure 1, TRIAL includes four knowledge-bases. From the viewpoint of construction, they are classified into two parts. One part consists of a statute knowledge-base and a theory knowledge-base, while the other part consists of a precedent knowledge-base and a new case knowledge-base. We adopted the following methods in designing knowledge-bases:

- statute knowledge-base and theory knowledge-base:

  In the example, statute and theory knowledge could be modeled in sufficient detail in advance. This enabled the fairly stable design of these knowledge-bases and provided a *framework* for the problem domain.

- **precedent knowledge-base and new case knowledge-base:**

  On the other hand, since precedents and new cases are partial information, designers of these knowledge-bases cannot model them in sufficient detail in advance. The design work cannot progress without interaction with other knowledge-bases.

  In these knowledge-bases, there are two ways of representing a newly introduced object: an object term and the value of a property of another object. For unstable objects, it is better to adopt the latter way, because properties are treated as a set of subsumption constraints, allowing lacking or incongruous properties to be abduced as a part of the answer.

The above methods can be thought of as guidelines when modeling knowledge in *QUIXOTE*. Knowledge-bases in the TRIAL system are modeled according to theses guidelines:

1. At first, the stable part of a series of knowledge is modeled and a framework for the problem domain is constructed.
2. Then, using this framework, the other part is constructed. Newly introduced objects are added as properties of objects in the framework.

By adopting these guidelines, users can retrieve important suggestions from knowledge-bases through interaction, without excessive examination of the overall, likely enormous, knowledge-base.

Refining a Model

Even if designers of new precedents adopt the above guidelines, when storing precedents in the database, they must still check the rule behavior with existing knowledge-bases.

In *QUIXOTE*, a 'trial and error' environment helps to refine a knowledge model. New model behavior can be examined by setting submodule relations between existing knowledge modules, which use the hypothetical reasoning mechanism, and rolling them back with the nested transaction management function. If a new model lacks some information, existing knowledge-bases assist in finding it by means of a hypothesis generation mechanism. This method enables rapid conformance between knowledge-bases as well as a model refinement process.

4.3 Implementing an Example Knowledge-base

In this section, we show how our example knowledge-base is implemented in TRIAL (as modules in *QUIXOTE*). We also show related query interaction.

The new case is represented as follows:

```
new-case :: {new-case/[who=mary,
                       while=break,
                       result=heart-attack ];;
            relation[state=employ,employee=mary ]
                      /[affiliation=organization[name="S" ],
                        job→driver ]},
```

where ";;" is a delimiter between rules.

The statute and the theory of its application are as follows:

$$
\begin{aligned}
&\textit{labor-law} :: \{\ \textit{organization}\ [\textit{name}{=}X] \\
&\qquad /[\textit{responsible}{\rightarrow}\textit{compensation}\ [\textit{object}{=}Y,\ \textit{money}{=}\textit{salary}\ ]] \\
&\qquad\qquad \Leftarrow \textit{judge}\ [\textit{case}{=}C]\ /[\textit{judge}{\rightarrow}\textit{insurance}\ ], \\
&\qquad\qquad\quad \textit{relation}\ [\textit{state}{=}Z,\ \textit{employee}{=}Y] \\
&\qquad\qquad\qquad /[\textit{affiliation}{=}\textit{organization}\ [\textit{name}{=}X]] \\
&\qquad\qquad\quad \|\ \{C \sqsubseteq \textit{case}\}\}. \\
&\textit{theory} :: \{\ \textit{judge}\ [\textit{case}{=}X]/[\textit{judge}{\rightarrow}\textit{insurance}\ ] \\
&\qquad\qquad \Leftarrow \textit{judge}\ [\textit{case}{=}X]/[\textit{judge}{\rightarrow}\textit{job-causality}\ ], \\
&\qquad\qquad\quad \textit{judge}\ [\textit{case}{=}X]/[\textit{judge}{\rightarrow}\textit{job-execution}\ ] \\
&\qquad\qquad\quad \|\ \{X \sqsubseteq \textit{case}\}\}.
\end{aligned}
$$

The abstract interpretation precedent rules are abstracted from the original precedent knowledge-base rules by TRIAL and the parameterization method mentioned in section 3.4.

$$
\begin{aligned}
&\textit{case}_1 :: \{\ \textit{judge}[\textit{case}{=}X]/[\textit{judge}{\rightarrow}\textit{job-execution}\ ] \\
&\qquad \Leftarrow \textit{relation}[\textit{state}{=}Y,\textit{employee}{=}Z]/[\textit{cause}{=}X], X \\
&\qquad\quad \|\ \{X \sqsubseteq \mathtt{parm}.\textit{case},\ Y \sqsubseteq \mathtt{parm}.\textit{state},\ Z \sqsubseteq \mathtt{parm}.\textit{employee}\}\}. \\
&\textit{case}_2 :: \{\ \textit{judge}[\textit{case}{=}X]/[\textit{judge}{\rightarrow}\textit{job-causality}\ ] \\
&\qquad \Leftarrow X/[\textit{while}{=}Y,\ \textit{result}{=}Z], \\
&\qquad\quad \|\ \{X \sqsubseteq \mathtt{parm}.\textit{case},\ Y \sqsubseteq \mathtt{parm}.\textit{while},\ Z \sqsubseteq \mathtt{parm}.\textit{result}\}\}.
\end{aligned}
$$

The object 'parm' represents the abstraction parameters. This object results from the abstraction of precedents. It is used to control judgment prediction. Its properties are defined as follows:

$$
\begin{aligned}
&\textit{parm} :: \{\ \mathtt{parm}/[\textit{case}{=}\textit{case},\ \textit{state}{=}\textit{relation},\ \textit{while}{=}\textit{job}, \\
&\qquad\qquad \textit{result}{=}\textit{disease},\ \textit{employee}{=}\textit{person}\ ]\}.
\end{aligned}
$$

To prevent over-abstraction, these values restrict the range of the variables X, Y, and Z in both precedent rules.

To enable the use of 'parm' for case$_1$ and case$_2$, we define the following submodule relation:

$$\textit{parm} \sqsupseteq_S \textit{case}_1 \cup \textit{case}_2.$$

This information is dynamically defined in the rule abstraction step, because the choice of precedents is made on an experimental basis.

The knowledge-base has the following subsumption relations:

case	$\sqsupseteq$	*new-case,*	*person*	$\sqsupseteq$	*mary,*
relation	$\sqsupseteq$	*employee,*	*job-causality*	$\sqsupseteq$	*insurance,*
disease	$\sqsupseteq$	*heart-attack,*	*job-execution*	$\sqsupseteq$	*insurance,*
job	$\sqsupseteq$	*break.*			

We can now query the knowledge-base:

1. According to the past precedents and theory, what kind of judgment can we predict for the new case?

?- *new-case* : *judge* [*case*=*new-case*]/[*judge*=*X*];;
   new-case $\sqsupseteq_S$ *parm* $\cup$ *theory*.

Note that the module *parm* is defined to inherit the abstracted precedent rules. Thus, we get three answers, in which the first is returned unconditionally, and the last two include hypotheses:

(a) $X \sqsubseteq$ *job-causality*,

(b) if *new-case* : *judge* [*case*=*new-case*] has *judge* $\sqsubseteq$ *job-execution*, then $X \sqsubseteq$ *insurance*,

(c) if *new-case* : *relation* [*state*=*employ*,*employee*=*mary*] has *cause*=*new-case*, then $X \sqsubseteq$ *insurance*.

2. According to labor law, what responsibility does Mary's company have?

?- *new-case* : *organization* [*name*="*S*"]/[*responsible*=*X*];;
   new-case $\sqsupseteq_S$ *parm* $\cup$ *labor-law*.

Note that, before issuing this query, the module *new-case* is already a submodule of module *theory* as a result of the previous query. Thus, two answers are returned with a generated hypotheses:

(a) if *new-case* : *judge* [*case*=*new-case*] has *judge* $\sqsubseteq$ *job-execution*,
then $X \sqsubseteq$ *compensation* [*obj*=*mary*, *money*=*salary*].

(b) if *new-case* : *relation* [*state*=*employ*, *employee*=*mary*] has
cause=*new-case*,
then $X \sqsubseteq$ *compensation* [*obj*=*mary*, *money*=*salary*].

QUIXOTE returns explanations (derivation graphs) with corresponding answers to TRIAL. The TRIAL user interface displays this graphically if so requested by the user. Judging an answer from the validity of the generated hypotheses and the corresponding explanation, the user can also update the database or change its abstraction strategy.

4.4 Other Useful Features

Some other features of *QUIXOTE* are also useful to the TRIAL system.

- A property inheritance mechanism enables the reduction of the amount of knowledge descriptions.

- A rule can be designated so as not to generate any hypothesis. Among the example knowledge-bases of TRIAL, rules in the statute and theory knowledge-bases are designated as such.

- Various browsing commands are supported. For example, a module before/after the saturation of rule inheritance can be displayed.

5 Concluding Remarks

We have discussed the representation of legal knowledge, the construction of a legal knowledge-base, and its refinement. We illustrated these steps with an example from our experimental legal reasoning system, TRIAL, which is based on the *QUIXOTE*.

As was clarified by this study, we focused on the following three main features of those supported by *QUIXOTE* to construct legal knowledge-bases:

1. *Knowledge modularization,*
2. *Hypothetical reasoning,* and
3. *Hypothesis generation.*

To expand the processing power of *QUIXOTE* to allow its application to a variety of knowledge information applications, we plan the following extensions:

- *Negation*: Traditional logic programming addresses both negation-as-failure (NAF) and classical negation. *QUIXOTE* considers various forms of negation: NAF and classical negation of an object term, negation of a subsumption constraint (property of an object term), and negation of a subsumption relation. Actually, NAF of an object term is already supported by *QUIXOTE*. And, we are planning to introduce classical negation of an object term with a restricted form of subsumption constraint negation, because subsumption constraints with negation are generally undecidable.
- *Meta-operation*: In TRIAL, we use a special object, parm, for abstracting and specializing rules. This is not desirable, however, for handling partial information. So, we plan to introduce some meta-operations such as the dynamic reduction of subgoals. In [3], Hamfelt *et al* describes the logical foundation of metaprogramming typically needed to represent legal knowledge. Although *QUIXOTE* has its own semantics of computation, it lacks the semantics of meta-operation. Enhancing the semantics for meta-operation like that in [3] should be considered.
- *Locality*: In designing knowledge-bases, it is very important to decide what is global and what is local. For example, in the current implementation of *QUIXOTE*, object identity and subsumption relations are global, while the existence of an object and subsumption constraint are local. We plan to make these definitions more flexible to strengthen the representation capability of *QUIXOTE*.
- *Visualization*: To support knowledge-base constructions, visualization of the knowledge-base is quite important. We plan to enhance *QUIXOTE*'s graphical user interface, e.g., by adding a hypertext style for operations such as searching and information filtering.
- *Heterogeneous constraints*: Legal reasoning presents various constraints besides subsumption, such as algebraic constraints. We plan to extend *QUIXOTE* into a heterogeneous, distributed, cooperative knowledge-base and problem solving environment to solve such constraints.

We began the design of the *QUIXOTE* in 1990 and have implemented several versions of the system. *QUIXOTE*, which runs in a UNIX environment, has been released as ICOT free software.

Acknowledgments

We would like to thank Nobuichiro Yamamoto (Hitachi, Ltd.) for TRIAL system design and implementation, and all the members of the *QUIXOTE* project for their valuable advice.

References

[1] H. Aït-Kaci, "An Algebraic Semantics Approach to the Effective Resolution of Type Equations," *Theoretical Computer Science*, no.45, 1986.

[2] K. D. Ashley, "Reasoning with Cases and Hypotheticals in HYPO," *Int. Journal of Man-Machine Studies*, 34(6), pp. 753-796, June, 1991.

[3] J. Barklund and A. Hamfelt, "Metaprogramming for Representation of Legal Principles," *Uppsala University Technical Report*, No. 61, July, 1990.

[4] A. J. Bonner and M. Kifer, "Transaction Logic Programming," *Proc. Int. Logic Programming*, 1993.

[5] S. Ceri, K. Tanaka, and S. Tsur (eds.), *Deductive and Object-Oriented Databases, (Proc. the Third Int. Conference on Deductive and Object-Oriented Databases (DOOD '93))*, LNCS 760, Springer, 1993.

[6] C. Delobel, M. Kifer, and Y. Masunaga (eds.), *Deductive and Object-Oriented Databases, (Proc. the Second Int. Conference on Deductive and Object-Oriented Databases (DOOD '91))*, LNCS 566, Springer, 1991.

[7] R. Demolombe, L. F. del Cerro and T. Imielinski (eds.), *Proc. Workshop on Nonstandard Queries and Answers*, Toulouse, July, 1991.

[8] T. F. Gordon, "An Abductive Theory of Legal Issues," *Int. Journal of Man-Machine Studies*, 35(1), pp. 95-118, July, 1991.

[9] O. Hødnebø and E. Løkketangen, "The Use of an ATMS in Consistency Checking of a Legal Expert System," *Proc. the Fourth Int. Conference on Artificial Intelligence and Law*, pp. 72-75, ACM Press, June, 1993.

[10] I. Jacobson, *et. al.*, "Object Oriented Software Engineering," *Addison and Wesley*, 1992.

[11] L. O. Kelso, "Does the Law Need a Technological Revolution?," *Rocky Mt. Law Rev.*, vol.18, pp.378-392, 1946.

[12] M. Kifer, G. Lausen, and J. Wu, "Logical Foundations of Object-Oriented and Frame-Based Languages," *SUNY TR 93/06*, June, 1993.

[13] D. Miller, "A Theory of Modules for Logic Programming," *The Int. Symposium on Logic Programming*, 1986.

[14] J. Rumbaugh, *et al.*, "Object-Oriented Modeling and Design," *Prentice-Hall*, 1991.

[15] S. Shlaer and S. Mellor, "Object-Oriented Systems Analysis," *Prentice-Hall*, 1988.

[16] C. Takahashi, K. Yokota, "A Legal Reasoning System on a Deductive Object-Oriented Database," *Proc. 5th Int. Hong Kong Computer Society Database Workshop*, Kow Loon, Hong Kong, February, 1994.

[17] N. Yamamoto, "TRIAL: a Legal Reasoning System (Extended Abstract)," *Joint French-Japanese Workshop on Logic Programming*, Renne, France, July, 1991.

[18] H. Yasukawa, H. Tsuda, and K. Yokota. "Objects, Properties, and Modules in $\mathcal{QUIXOTE}$," *Proc. Int. Conf. on FGCS*, ICOT, Tokyo, June 1-5, 1992.

[19] H. Yasukawa and K. Yokota, "Labeled Graph as Semantics of Objects," *Proc. Joint Workshop of SIGDBS and SIGAI of IPSJ*, Nov., 1990.

[20] K. Yokota and M. Shibasaki, "Can Databases Predict Legal Judgements?" *Joint Workshop of IPSJ SIGDSS and IEICE SIGDE (EDWIN)*, Nagasaki, July 21-23, 1993. (in Japanese)

[21] K. Yokota, H. Tsuda, and Y. Morita, "Specific Features of a Deductive Object-Oriented Database Language $\mathcal{QUIXOTE}$," *Workshop on Combining Declarative and Object-Oriented Databases, (ACM SIGMOD '93 Workshop)*, Washington DC, May 29, 1993.

[22] K. Yokota and H. Yasukawa, "Towards an Integrated Knowledge-Base Management System — Overview of R&D on Databases and Knowledge-Bases in the FGCS Project," *Proc. Int. Conf. on FGCS*, ICOT, Tokyo, June 1-5, 1992.