ICOT Technical Memorandum: TM-1319

TM-1319

Constructing a Legal Database on Quixote

by
C. Takahashi (JIPDEC) & K. Yokota

January, 1995

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191 ~5

Institute for New Generation Computer Technology

Constructing a Legal Database on QUIXOTE

Chie Takahashi

Japan Information Processing
Development Center(JIPDEC)
Kikaishinkou-Kaikan, 3-5-8,
Shibakoen, Minato-ku, Tokyo 105,
JAPAN.

e-mail: j-takaha@icot.or.jp.

Kazumasa Yokota

Institute for New Generation
Computer Technology (ICOT)
21F., Mita-Kokusai Bldg., 1-4-28,
Mita, Minato-ku, Tokyo 108,
JAPAN.

e-mail: kyokota@icot.or.jp.

Abstract

Legal reasoning is one application of large-scale knowledge information processing, where artificial intelligence, natural language processing, databases and other technologies are integrated. It is the target for the next-generation of databases. In order to investigate whether or not the deductive object oriented database (DOOD) language/system QUIXOTE is effective in legal reasoning, we are both writing the United Nations Convention on Contracts for the International Sale of Goods (CISG) in QUIXOTE and constructing the legal database on QUIXOTE at ICOT. In this paper, we show that QUIXOTE is suitable for constructing legal databases requiring knowledge representation and for debugging them.

1 Introduction

Legal reasoning is one application of large-scale knowledge information processing where artificial intelligence, natural language processing, databases and other technologies are integrated.

From the database point of view, legal reasoning requires access to vast data and knowledge sources such as written law (for example, constitution, decrees, orders, ordinaces, regulations and so on) and unwritten law (for example, customary law, case law, theories, social norms, industrial policies and so on). Specially, we have a great many precedents. Because it is diffucult for a person to deal with such amounts of legal data and knowledge, it is hoped that databases can support powerfully legal reasoning. But, it is said that conventional legal databases where laws and precedents are stored are not so useful and even that it is possible to find correct legal data and knowledge in law faster than the databases. The reason is that the databases store laws and precedents written in natural language and are used as document retrieval systems. We show a way to construct a legal database more powerful than the conventional ones.

To make powerful legal databases, it is important to consider what kinds of data and knowledge are abstracted from raw data, and how to represent legal abstract knowledge.

The characteristics of legal data and knowledge are as follows:

- A variety of data and knowledge such as laws, precedents, jurisprudential theories, and background knowledge.
- Complex data structures such as semantic networks and concept taxonomy.
- Ambiguity of the semantics or a variety of interpretations of sentences and cases.
- Partiality of information as in cases and interpretation implied by sentences.
- A large volume of data and knowledge.

As they are similar to natural scientific databases, we must cope with many difficulties in constructing legal databases.

This paper focusses on constructing legal databases. We show how to represent legal data and knowledge and how to create legal databases by resolving conflicting information and complementing missing information.

In Japan's FGCS (Fifth Generation Computer System) project, we have designed and developed QUIXOTE, a deductive object-oriented database (DOOD) language/system [8, 5, 10, 11, 12]. It is a knowledge representation language, which plays an important role in knowledge information processing requiring a high capability of representation and query processing. At present, we are constructing a legal database using the United Nations Convention on Contracts for the International Sale of Goods (CISG) and using it to investigate how to represent legal knowledge, and how effective QUIXOTE is in legal reasoning. In this paper, we introduce a method of writing and debugging a legal database using CISG.

Section 2 briefly introduces CISG and gives that part of CISG explained in this paper. Section 3 gives an outline of QUIXOTE used in that part of

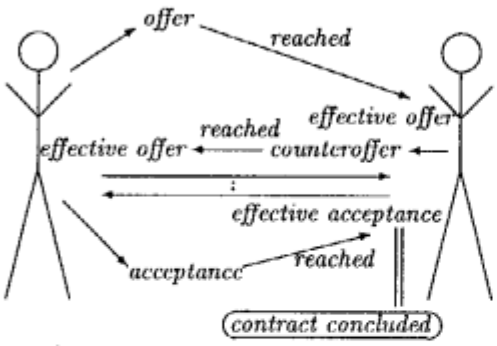

Figure 1: A Simplified Legal Model of the Articles

CISG. Section 4 introduces a method of writing and debugging a legal database using CISG.

2 CISG Part II. FORMATION OF THE CONTRACT

CISG is the acronym for the *United Nations Convention on Contracts for the International Sale of Goods*, which has 101 articles and is also known as the *Vienna Sales Convention*. Domestic trade follows domestic law, while international trade observes this convention. It was adopted at a conference for diplomats in Vienna on April 11, 1980, and took effect on January 1, 1988.

We are constructing a legal database using *Part II. FORMATION OF THE CONTRACT*, which is the core of CISG. In this paper, we show the part of CISG Part II. that is necessary to conclude a contract.

In Section 2.1, we briefly explain a simplified legal model of the articles which are part of CISG Part II. shown in Appendix A. In Section 2.2, we list common questions about CISG Part II.

2.1 A Simplified Legal Model of the Articles

The following briefly explains a simplified legal model of the articles in Appendix A. Figure 1 shows the model. The words shown in figure 1 form the constituent activities in CISG Part II.

To conclude a contract, two persons are necessary. One person makes an *offer* to the other person. If the *offer reaches* the offeree, it becomes *effective*. The offeree may make a *counteroffer* to the *effective offer* or give an *acceptance* of the *effective offer* to the other person. If the *acceptance reaches* the addressee, it becomes *effective* and a *contract* is concluded.

2.2 Common Questions about CISG Part II.

These are common questions about CISG Part II.

- Does a proposal constitute an *offer*?
- Does a reply to an *offer* constitute a *counteroffer*?
- Does an *offer* become *effective*?
- Does a reply to an *offer* constitute an *acceptance*?
- Does an *acceptance* become *effective*?
- Is a *contract* concluded?

It is important that the above questions be represented easily.

3 An Outline of QUIXOTE

QUIXOTE is based on several concepts: object identity, subsumption relations, subsumption constraints, property inheritance, modules, submodule relations, and rule inheritance. From a database point of view, the language is a DOOD language while, from a logic programming point of view, it is thought of as an extended constraint logic programming language based on subsumption constraints. There are some differences from the new F-logic[2, 3]: the representation of object identity, the introduction of subsumption constraints, update semantics, and query processing. This section explains some of its features used in the above articles. For details on QUIXOTE, see [10, 11, 12].

3.1 Object Identity and Subsumption Relation

Objects in QUIXOTE are identified by extended terms called *object terms* that correspond to object identifiers (oids). An object term consisting of an atomic symbol is classified as *basic*, while an object term in the form of a tuple is referred to as *complex*. For example, *apple*, *fruit*, *parallelogram* and *quadrangle* are basic, while $apple[color = green]$ is complex. Generally, an object term is a variable or a term having the following form:

$$o[l_1 = t_1, \cdots, l_n = t_n] \ (0 \leq n)$$

where $o, l_1, \cdots, l_n$ are basic and $t_1, \cdots, t_n$ are object terms. $l_1, \cdots, l_n$ are called *labels*.

Object terms are related to each other by a *subsumption relation* (written $\sqsubseteq$, a kind of is_a relation). For example,

$$\begin{aligned} apple &\sqsubseteq fruit \\ parallelogram &\sqsubseteq quadrangle \end{aligned}$$

Given partial order between the basic object terms, it is extended between complex object terms as usual:

$$apple[color = green] \sqsubseteq apple.$$

As the construction of a lattice from a partially ordered set, like that in [1], is well-known, we can assume that a set of object terms with *top* and *bot* constitutes a lattice, without losing generality. The meet and join operations of o_1 and o_2 are denoted by $o_1 \downarrow o_2$ and $o_1 \uparrow o_2$, respectively.

3.2 Subsumption Constraints

A property of an object is represented as a subsumption constraint. The pair constituted by an object term o and a label l, denoted $o.l$, is called a *dotted term*, which plays the role of a variable ranging over the domain of the object terms. Furthermore, a pair constituted by a dotted term and a label is itself also a dotted term. If t_1, t_2 is an object term or dotted term, then a *subsumption constraint* is defined as follows:

$$t_1 \sqsubseteq t_2.$$

That is, a *property* of an object o is a subsumption constraint with a dotted term starting with o. In other words, it is defined as a triple constituted by a label, a subsumption relation, and a value. For example, the *shape* of *apple* is represented by the following subsumption constraint: $apple.shape \cong round$.

The syntactic construct for representing an object term with subsumption constraints is called an *attribute term*. Let o be an object term, C a set of subsumption constraints, then $o|C$ is an attribute term. For example, the following attribute term represents that the *size* of *apple* is *small*, the *shape* of *apple* is *round* and the *sort* of *apple* is *fruit*:

$$apple|\{apple.size \cong small,\ apple.shape \cong round,\ apple.sort \cong fruit\}$$

There are some syntax sugars in QUIXOTE:

$$o|\{o.l \sqsupset t\} \Leftrightarrow o/[l \rightarrow t]$$
$$o|\{o.l \sqsupseteq t\} \Leftrightarrow o/[l \leftarrow t]$$
$$o|\{o.l \cong t\} \Leftrightarrow o/[l \rightarrow\!\!\!\!\!- t]$$

Thus, the above attribute term can be represented as follows:

$$apple/[size - small, shape - round, sort - fruit]$$

Notice that there are two kinds of properties: those that appear in object terms and those as subsumption constraints. The former are *intrinsic* for an object, while the latter are *extrinsic* for an object. Regarding the extrinsic properties, the following constraint solver is applied to a set of subsumption constraints:

$$\begin{aligned}
o = o &\implies \text{eliminated} \\
o_1 \sqsupseteq o_2 &\longrightarrow o_2 \sqsubseteq o_1 \\
o_1 \sqsubseteq o_2, o_2 \sqsubseteq o_3 &\implies o_1 \sqsubseteq o_3 \\
o_1 \not\perp o_2, o_1 \not\perp o_3 &\implies o_1 \not\perp o_2 \downarrow o_3 \\
o_1 \sqsubseteq o_3, o_2 \sqsubseteq o_3 &\implies o_1 \uparrow o_2 \sqsubseteq o_3
\end{aligned}$$

Any set of subsumption constraints will produce a unique solution by applying the above rules[9].

3.3 Rule and Module

A rule is defined as in constraint logic programming, as follows:

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

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*, $a_1, \cdots, a_n, D$ is called a *body*, and a_i is called a *subgoal*. A rule means that if the body is satisfied then the head is satisfied. If a body is empty, then the rule is called a *fact*.

For example, the following is a rule for a quadrangle.

$$\begin{aligned}
&quadrangle/[figure \rightarrow parallelogram] \\
&\quad \Leftarrow quadrangle/[angle1 = A, angle2 = B, \\
&\qquad angle3 = C, angle4 = D] \\
&\quad || \{A = C, B = D\}.
\end{aligned}$$

It means that if all faced angles of a quadrangle are equal, then the quadrangle is a parallelogram.

A module corresponds to a part of the world (situation) or a local database. The module concepts play an important role in classifying knowledge, modularizing a program or a database, assumption-based reasoning, and dealing with any inconsistency in a database in QUIXOTE.

A *module* is defined as a set of rules as follows:

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

where m is an object term called a *module identifer* (mid) and $r_1, \cdots, r_n$ are rules. m is sometimes used instead of a module itself, if there is no confusion.

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

$$m_0 :: \{a_0 \Leftarrow m_1 : a_1, \cdots, m_n : a_n || D\}$$

where $m_0, m_1, \cdots, m_n$ are mids. It means that the module m_0 has a rule such that if a_i and D are satisfied in the module m_i for all $1 \leq i \leq n$, then a_0 is satisfied in the module m_0. As an attribute term can be separated into an object term and a set of constraints, the rule can be rewritten as follows:

$$m_0 :: \{o_0 | C_0 \Leftarrow m_1 : o_1, \cdots, m_n : o_n || C\}$$

where $a_i = o_i | C_i (0 \leq i \leq n)$ and $C = C_1 \cup \cdots \cup C_n \cup D$.

Importing and exporting rules are done by *rule inheritance*, defined in terms of the binary relation (written $\sqsupseteq_S$) between modules, called a *submodule relation* as follows:

$$m_1 \sqsupseteq_S m_2,\ \ m_1 :: R_1,\ \ m_2 :: R_2 \implies m_1 :: R_1 \cup R_2,\ \ m_2 :: R_2$$

where m_1, m_2 are modules and R_1, R_2 are sets of rules. The right hand side of $\sqsupseteq_S$ in a submodule definition may be a formula of mids with set operations.

It is possible for inconsistent knowledge to coexist by making use of the module mechanism. For example, consider that it cannot be said for certain whether the *taste* of fruit is *sweet* or *sour*. The following shows how such a problem is handled:

$$m_1 :: fruit/[taste = sweet].$$
$$m_2 :: fruit/[taste = sour].$$

where m_1 and m_2 are not related by submodule relation.

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

3.4 Query Processing

Query processing basically corresponds to resolution and constraint solving in constraint logic programming. One of the main features of data and knowledge in knowledge information processing, such as legal reasoning, is that the information is partial. That is to say, sufficient information need not necessarily be given. For example, a new case might lack some important facts. So a query and an answer are extended for treating partial information.

A *query* is defined as the pair (A, P) (written ?-A;;P) of a set of attribute terms A and a program P, where A is referred to as the *goal* and P as a *hypothesis*.

Consider a database DB. A query ?-A;;P to DB is equivalent to a 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 the details in [12].

An *answer* is defined as the triple (D, V, E) of a set of subsumption constraints D that cannot be solved during query processing, a set of variable constraints V that are bounded during query processing, and the corresponding derivation flow E. D lacks information in the database, obtained by abduction, V is an answer in the sense of constraint logic porgramming and E is the explanation.

4 Constructing a Legal Database on QUIXOTE

In this section, we explain how to construct a legal database on QUIXOTE using *CISG Part II. FORMATION OF THE CONTRACT.*

Because sentences in laws are very complex, it is very important to represent a model of laws efficiently and flexibly. Beyond that, to construct legal databases, it is necessary to provide a verification and debugging environment because a constructor may make mistakes in the syntax of a knowledge representation or in constructing a model of law, or a semantic model of a knowledge representation may differ from a semantic model of law. As in legal knowledge, there are some ambiguity and partiality.

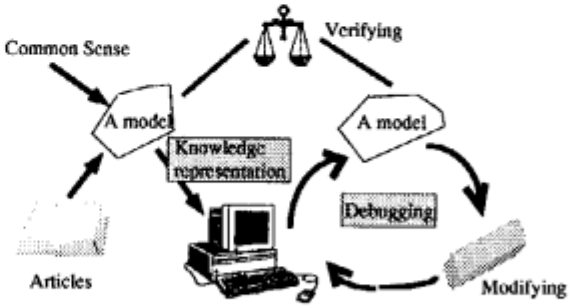

Figure 2: Constructing a Legal Database

Section 4.1 explains object terms and modules in CISG Part II. Section 4.2 explains the articles in QUIXOTE. Section 4.3 shows how a legal database is debugged using QUIXOTE query processing.

4.1 Objects and Modules in CISG Part II.

An article usually constitutes a legal concept when an incident meets some conditions imposed by the article. So at first we consider incidents in the real world and defined legal concepts in CISG Part II.

The following are incidents in the real world, defined legal concepts, and the articles that constitute the legal concepts in Appendix A.

1) Incidents in the real world
   the offeror of a contract, the offeree of a contract, a proposal, a reply to the proposal, a reply to the reply of the proposal, ···

2) Legal concepts and the articles which constitute them
   an offer(Article 14(1)), a counteroffer for the offer, a counteroffer for the counteroffer for the offer, ···(Article 19(1)), an effective offer(Article 15(1)), an acceptance(Article 18(1)), an effective acceptance(Article 18(2)), a concluded contract(Article 23), an offer reaches, a counteroffer reaches, ···, an acceptance reaches, ··· (Article 24)

Because the legal concepts in 2) can be devided between those related to effectiveness, which need some information about time, and the others, we consider 3 modules: module fact, def and effect_def added 1). We also consider object terms

that represent the above incidents and the above legal concepts.

The 3 modules and the object terms are as follows:

(1) module fact:
Module fact has object terms that represent an incident in the real world.

- offerorOfCon
- offereeOfCon
- proposal
- reply[order=1], ···, reply[order=N] [1]

(2) module def:
Module def has object terms that represent a legal concept except effective ones.

- offer[order=0], ···, offer[order=N] [2]
- acceptance[order=N]
- contract[order=N]
- reaches[order=0], ···, reaches[order=N]

(3) module effect_def:
Module effect_def has object terms that represent a legal effective concept.

- offer[order=0], ···, offer[order=N]
- acceptance[order=N]

When we examine the relations between the above object terms and the articles in Appendix A, we notice that each article defines a correspondence between objects (Figure 3).

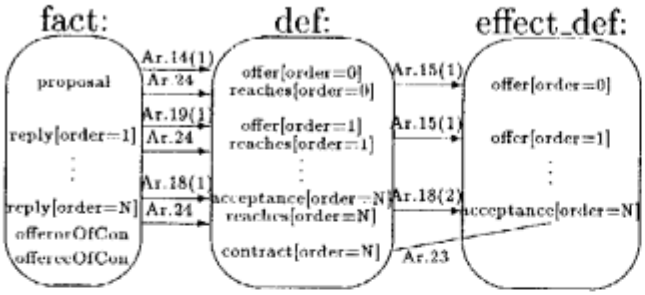

Figure 3: object terms and modules in *QUIXOTE*

def::offer[order=N] represents that the Nth reply, which exists in the real world, constitutes an offer according to CISG. In the same way,

effect_def::offer[order=N]
def::reaches[order=N]
def::acceptance[order=N]
def::contract[order=N]

represent that Nth reply in the real world constitutes
an effective offer
an reached reply
an acceptance
a concluded contract

respectively according to CISG.

The common questions introduced in Section 2.2 are represented in *QUIXOTE* as follows:

- Does a proposal constitute an *offer*?
  ?- def:offer[order=0].
- Does a reply to an *offer* constitute an (*counter-*) *offer*?
  ?- def:offer[order=K]. (1 ≤ K ≤ N)
- Does an *offer* become *effective*?
  ?- effect_def:offer[order=K]. (1 ≤ K ≤ N)
- Does a reply to an *offer* constitute an *acceptance*?
  ?- def:acceptance[order=N].
- Does an *acceptance* become *effective*?
  ?- effect_def:acceptance[order=N].
- Is a *contract* concluded?
  ?- def:contract[order=N].

4.2 Articles in *QUIXOTE*

In Appendix B, the articles in Appendix A are written in *QUIXOTE* using the object terms and the modules in Section 4.1. When we considered descriptions of the articles in *QUIXOTE*, we tried to make the descriptions represent the articles with regard to the legal model in Section 2.1. Because an article usually connects an incident to a legal concept, we also tried to make them represent conditions in the articles as intrinsic properties of object terms in module fact, not in module def nor in module effect_def.

For example, the article 18 (1) in Appendix A:

> A statement made by or other conduct of the offeree indicating assent to an offer is an acceptance. Silence or inactivity does not in itself amount to acceptance.

is represented by the rule in Appendix B:

```
def::acceptance[order=N] ⇐
    effect_def:offer[order=N-1] /[offeree=A],
    fact:reply[order=N]
            /[act → indication_of_intention,
              intention=assent, addresser=A];;
```

The rule means that

> When the N-1th reply of which offeree is A and which exists in the real world has become effective as an offer, and the Nth reply of which addresser is A is a conduct indicating assent to the offer, the Nth reply constitutes an acceptance.

[1] reply[order=1] represents a reply to a proposal and reply[order=K] represents a reply to reply[order=K-1] (1 < K ≤ N).

[2] offer[order=K] represents a counteroffer for offer[order=K-1] (1 < K ≤ N).

4.3 Modifying the Legal Database

In constructing legal databases, debugging is necessary because a constructor of the legal database may make mistakes in the syntax of a knowledge representation or in constructing a model of law, or the semantic model of the knowledge representation may differ from the semantic model of law.

We can debug a legal database using *QUIXOTE* query processing, because it returns an answer with hypotheses.

The following are two examples of debugging the legal database in Appendix B. One is an exception representation of law, and the other is an abstract representation of law.

(1) Exception representation of Law

As you can see from Figure 1, a reply to an offer cannot constitute a counteroffer and an acceptance at the same time. So, the expected answer of the following question is NO.

```
?- def:acceptance[order=N], def:offer[order=N].
```

But if we ask the above question to the legal database shown in Appendix B, we get the following answer with a hypothesis.

IF
fact:reply[order=N [3]].containing_modifications ≅ yes
fact:reply[order=N].act ⊑ indication_of_intention
fact:reply[order=N].intention ≅ assent
THEN YES

This answer indicates that the reply, which meets the above hypothesis, accepts an offer and rejects the offer. So we notice that we must modify the rules of Article 18(1) and Article 19(1) which define def:acceptance[order=N] and def:offer[order=N] respectively. Because the above hypothesis is the condition of a *counteroffer*, we must modify only the rule of Article 18(1) which defines an *acceptance*. That is, we add the attribute "containing_modifications=no" to the rule as follows:

```
def::acceptance[order=N] ⇐
    effect_def:offer[order=N-1]/[offeree=A],
    fact:reply[order=N]
              /[act → indication_of_intention,
                intention=assent, addresser=A,
                containing_modifications=no];;
```

If we modify the rule of Article 18(1) in the above way and ask the above question

```
?- def:acceptance[order=N], def:offer[order=N].
```

we get the expected answer, NO.

[3]We actually get all answers that consist of reply[order=K] ($1 \leq K \leq N$) in module fact.

(2) Abstract Representation of Law

If we ask the legal database shown in Appendix B the following question:

```
?- effect_def:acceptance[order=N].
```

we get NO. It is an unexpected answer. So, in order to find the cause of the unexpected answer, we ask the legal database the following questions, which define the effect_def:acceptance[order=N] in the rule of Article 18(2).

```
?- def:acceptance[order=N].
?- def:reaches[order=N].
```

We get the following answers:

IF
fact:reply[order=N].containing_modifications ≅ no
fact:reply[order=N].act ⊑ indication_of_intention
fact:reply[order=N].intention ≅ assent
THEN YES

and

IF
fact:offerorOfCon.having_a_place_of_business_or_mailing_address ≅ no
fact:reply[order=N].act ≅ delivery[place=habitual]
THEN YES
IF
fact:offerorOfCon.having_a_place_of_business_or_mailing_address ≅ yes
fact:reply[order=N].act ≅ delivery[place=mailing_address]
THEN YES
IF
fact:offerorOfCon.having_a_place_of_business_or_mailing_address ≅ yes
fact:reply[order=N].act ≅ delivery[place=business]
THEN YES
IF
fact:reply[order=N].act ≅ delivery[place=personal]
THEN YES
IF
fact:reply[order=N].act ≅ oral_conduct
THEN YES

So that we do not get NO as the answer of ?- effect_def:acceptance[order=N], the above answers of ?- def:acceptance[order=N] and ?- def:reaches[order=N] must have hypotheses in common. Then, by paying attention to constraints of fact:reply[order=N].act, we notice that

$$\begin{aligned} \text{oral\_conduct} &\sqsubseteq \text{indication\_of\_intention} \\ \text{delivery} &\sqsubseteq \text{indication\_of\_intention} \end{aligned}$$

are necessary.

If we ask the legal database to which are added the above subsumption relations about

?- effect_def:acceptance[order=N], we get 5 answers whose hypotheses combine the hypotheses of the answers of ?- def:acceptance[order=N], and ?- def:reaches[order=N] as in the following answer:

IF
fact:offerorOfCon.having_a_place_of_business_or_mailing_address ≅ no
fact:reply[order=N].act ≅ delivery[place=habitual]
fact:reply[order=N].containing_modifications ≅ no
fact:reply[order=N].intention ≅ assent
def:reaches[order=N].kind ⊑ within_a/the_time [4]
THEN YES

5 Concluding Remarks

Although legal reasoning is one of the most attractive applications for next generation databases, there have not been many studies involving legal databases. In this paper, we represent legal data and knowledge in the framework of a DOOD and show how to reduce ambiguity and conflict in the databases as an application of query processing. Our contributions in this paper can be summarized as follows:

1. We show that *QUIXOTE*, as a DOOD language /sytem, is effective for representing legal data and knowledge with ambiguity and conflict.

2. We show that query capabilities, such as conditional query and hypothesis generation, work effectively for debugging legal databases.

This paper explains the above using specific examples from CISG Part II. As *QUIXOTE* is a deductive object-oriented database system, it can manage legal data and knowledge efficiently as explained in this paper. In addition, the query capabilities are effective in legal reasoning[6]. A legal database on *QUIXOTE* is thus more powerful than a conventional one.

There are some studies which represent law. For example, [7] which represents the British Nationality Act is notable. The main differences from our study are in the use of predicate representation and the attempt to use non-monotonic reasoning. One of the characteristics of legal data and knowledge is the partiality of information as in cases and interpretation implied by sentences. Because *QUIXOTE* can deal with partiality of information [6], while predicate representation cannot, *QUIXOTE* can represent law more naturally than [7]. As non-monotonic reasoning is very important, we are considering extending *QUIXOTE* to be able to deal with non-monotonic reasoning.

[4] If a function that evaluates time constraint, explained in Section 5, is introduced in *QUIXOTE*, it is replaced by the constraints of object terms of module fact.

Moreover, in order to create a bigger legal database, we plan the following experiments and extensions:

(1) Modal Representation

Modal representations in which *can*, *must* or *may* is used often appear in law. We are trying to describe the modal representations in *QUIXOTE* using modules. If we can describe them, we may create a powerful legal database system for CISG that shows the rights and obligations of the seller, the buyer or the contractor, and simulates, for example, a negotiation for concluding a contract when the person concerned in the contract exercises his rights.

(2) Expansion of *QUIXOTE*

We are trying to improve *QUIXOTE* so that it can use external functions. For example, if *QUIXOTE* can use an external function that evaluates time constraint (written $\leq_{time}$ and $<_{time}$, for example, comparing two dates in order to find out which one is earlier and counting the number of days), we can describe module effect[time=T] in *QUIXOTE* as follows:

```
effect[time=T]::X ⇐
        effect_def:X/[the_date_of_beginning=S,
                      the_date_of_terminated=E]
        |{S≤time T, T<time E}.
```

The module effect[time=T] contains the object terms effective at time T.

(3) *Helios*

Although 'reasonable' is used in many articles of CISG, the meaning of the word is not defined in CISG. The reason is that the court wants to clarify the meaning without the influence of social and period changes. A powerful legal database system thus has a function that adopts human judgement.

Helios which is a heterogeneous, destributed, cooperative problem solving system, is being studied and developed at ICOT [14]. *Helios* can use any knowledge representation language, database, constraint solver, application program, or even a person as a problem solver, and can solve problems cooperatively using the problem solvers.

Using *Helios* with *QUIXOTE* as a problem solver, we may create a powerful legal database system for CISG on *Helios* which adopts human judgement.

We will expand the legal database using CISG to a very large database/knowledge base as we enhance *QUIXOTE* using *Helios*.

Acknowledgments

The authors wish to thank all the members of the *QUIXOTE* project, and Dr. Aiba and Dr. Nitta (ICOT) for their valuable advice and comments.

Appendix A: Part of CISG Part II.

Article 14

(1) A proposal for concluding a contract addressed to one or more specific persons constitutes an offer if it is sufficiently definite and indicates the intention of the offeror to be bound in case of acceptance. A proposal is sufficiently definite if it indicates the goods and expressly or implicitly fixes or makes provision for determining the quantity and the price.

Article 15

(1) An offer becomes effective when it reaches the offeree.

Article 18

(1) A statement made by or other conduct of the offeree indicating assent to an offer is an acceptance. Silence or inactivity does not in itself amount to acceptance.

(2) An acceptance of an offer becomes effective at the moment the indication of assent reaches the offeror. An acceptance is not effective if the indication of assent does not reach the offeror within the time he has fixed or, if no time is fixed, within a reasonable time, due account being taken of the circumstances of the transaction, including the rapidity of the means of communication employed by the offeror. An oral offer must be accepted immediately unless the circumstances indicate otherwise.

Article 19

(1) A reply to an offer which purports to be an acceptance but contains additions, limitations or other modifications is a rejection of the offer and constitutes a counteroffer.

Article 23

A contract is concluded at the moment when an acceptance of an offer becomes effective in accordance with the provisions of this Convention.

Article 24

For the purposes of this Part of the Convention, an offer, declaration of acceptance or any other indication of intention "reaches" the addressee when it is made orally to him or delivered by any other means to him personally, to his place of business or mailing address or, if he does not have a place of business or mailing address, to his habitual residence.

Appendix B: Articles in *QUIXOTE*

Article 14 (1)

```
def::offer[order=0]/[offeror=A, offeree=B] ⇐
  fact:proposal
    /[having_a_purpose_to_conclude_a_contract=yes,
      addressed_to_one_or_more_specific_persons=yes
      sufficiently_definite=yes,
      indicating_the_intention_of_the_offeror_
          to_be_bound_in_case_of_acceptance=yes,
    addressee=A, addresser=B];;
fact::proposal/[sufficiently_definite=yes] ⇐
  fact:proposal
    /[indicating_the_goods=yes,
      expressly_fixing_the_quantity_
          and_the_price=yes];;
fact::proposal/[sufficiently_definite = yes] ⇐
  fact:proposal
    /[indicating_the_goods=yes,
      implicitly_fixing_the_quantity_
          and_the_price=yes];;
fact::proposal/[sufficiently_definite=yes] ⇐
  fact:proposal
    /[indicating_the_goods=yes,
      making_provision_for_determining_
          the_quantity_and_the_price=yes];;
```

Article 15 (1)

```
effect_def::offer[order=N]
          /[the_date_of_becoming_effective=S,
            offeror=A, offeree=B] ⇐
  def:offer[order=N]/[offeror=A, offeree=B],
  def:reaches[order=N]/[the_date_of_reaching=S];;
```

Article 18 (1)

```
def::acceptance[order=N] ⇐
  effect_def:offer[order=N-1] /[offeree=A],
  fact:reply[order=N]
          /[act → indication_of_intention,
            intention=assent, addresser=A];;
```

Article 18 (2)

```
effect_def::acceptance[order=N]
          /[the_date_of_becoming_effective=S] ⇐
  def:acceptance[order=N],
  def:reaches[order=N]
          /[kind → within_a/the_time,
            the_date_of_reaching=S];;
```

Article 19 (1)

```
def::offer[order=N]/[offeror=A, offeree=B] ⇐
  effect_def:offer[order=N-1]/[offeree=A],
  fact:reply[order=N]
          /[act → indication_of_intention,
            intention=assent,
            containing_modifications=yes,
            addressee=A, addresser=B];;
```

Article 23

```
def::contract[order=N] ⇐
          effect_def:acceptance[order=N];;
```

Article 24

```
def::reaches[order=N]/[the_date_of_reaching=T] ⇐
  fact:reply[order=N]
    /[act=oral_conduct, the_date_of_oral_conduct=T];;
def::reaches[order=N]/[the_date_of_reaching=T] ⇐
  fact:reply[order=N]
    /[act=delivery[place=personal],
      the_date_of_delivery=T];;
def::reaches[order=N]/[the_date_of_reaching=T] ⇐
  fact:reply[order=N]
    /[act=delivery[place=business],
      the_date_of_delivery=T, addressee=R],
  fact:R/[having_a_place_of_business_or_
          mailing_address=yes] ||{R ⊑ human};;
def::reaches[order=N]/[the_date_of_reaching=T] ⇐
  fact:reply[order=N]
    /[act=delivery[place=mailing_address],
      the_date_of_delivery=T, addressee=R],
  fact:R/[having_a_place_of_business_or_
          mailing_address=yes] ||{R ⊑ human};;
def::reaches[order=N]/[the_date_of_reaching=T] ⇐
  fact:reply[order=N]
    /[act=delivery[place=habitual],
      the_date_of_delivery=T, addressee=R],
  fact:R/[having_a_place_of_business_or_
          mailing_address=no] ||{R ⊑ human};;
```

References

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

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

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

[4] S. Niibori, "The Uniform Law on International Sales — Vienna Sales Convention and Trade Contract—", Doubunkan Publishing Company, 1991. (in Japanese)

[5] T. Nishioka, R. Ojima, H. Tsuda and K. Yokota, "Procedural Semantics of a DOOD Programming Language *QUIXOTE*", *Joint Workshop of IPSJ SIGDSS and IEICE SIGDE (EDWIN)*, Nagasaki, July 21-23, 1993. (in Japanese)

[6] T. Nishioka, K. Yokota, C. Takahashi and S. Tojo, "Constructing a Legal Knowledge-base with Partial Information", *Proc. ECAI'94 Workshop on Artificial Normative Reasoning*, Aug.8, 1994.

[7] M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammondo and H. T. Cory, "The British Nationality Act as a Logic Program", COMMUNICATIONS OF THE ACM, Vol.29, No.5, pp 370-386, 1986.

[8] C. Takahashi, Y. Morita, T. Nishioka and H. Tsuda, "Features and Implementation of a Deductive Object-Oriented Database System *QUIXOTE*", *Joint Workshop of IPSJ SIGDSS and IEICE SIGDE (EDWIN)*, Nagasaki, July 21-23, 1993. (in Japanese)

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

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

[11] 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.

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

[13] 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)

[14] K. Yokota and A. Aiba, "A New Framework of Very Large Knowledge Bases", *Knowledge Building and Knowledge Sharing*, eds K. Fuchi and T. Yokoi, Ohmsha and IOS Press, 1994.