ICOT Technical Report: TR-0889

TR-0889

Query Processing for Partial Information Databases in QUIXOTE

by
K. Yokota, T. Nishioka (MRI), H. Tsuda
& S. Tojo (MRI)

August, 1994

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

Query Processing for Partial Information Databases in $\mathcal{QUIXOTE}$

Kazumasa Yokota Toshihiro Nishioka *
Hiroshi Tsuda Satoshi Tojo *
Institute for New Generation Computer Technology (ICOT)
Mita-Kokusai Bldg. 21F., 1-4-28, Mita, Minato-ku, Tokyo 108, Japan
e-mail: {kyokota,tsuda}@icot.or.jp

Abstract

In advanced knowledge processing, partial information plays an important role for coping with complex data and knowledge. To treat such information, we developed a knowledge representation language (which may be considered a deductive object-oriented database language) $\mathcal{QUIXOTE}$. The language has both features of logic and object-orientation concepts, as well as database features. In this paper, we introduce the query processing mechanism on partial information databases of $\mathcal{QUIXOTE}$, as a tool for effective data processing, taking an example of legal reasoning, and show its applicability to many applications in artificial intelligence.

1 Introduction

According to the recent increase of database applications, a great variety of data came to be used and the structures of knowledge representation became more and more complicated. Especially, those data types in *new* applications are often much different from the conventional business applications. In the Japanese Fifth Generation Computer System (FGCS) and its Follow-On projects [17], we have engaged in a variety of knowledge information processing systems such as natural language processing, legal reasoning, and genetic information processing. From this experience, we can conclude that one of the major features of data and knowledge in such new fields is *partiality of information*. For example, it is very difficult to define a common schema for a set of precedents in legal reasoning, or that for discourses in natural language processing because of factors including the ambiguity in natural language and the complexity of the knowledge itself. That is, in many cases, a database schema (or a predicate with a fixed number of arguments) cannot be defined in advance. Attributes only have indefinite values, such as constraints, and the data itself might be ambiguous or inconsistent. Partiality in this paper is different from *incompleteness* in databases, which have null values and information disjunction or negation. Partiality of information means that no one can specify *all* important information in sufficient detail in advance. Even if the schema could be fixed at some time, it might be frequently changed according to changes in the environment. Under environments such as knowledge information processing, representation of partial information is indispensable.

Inference of partial information is also important in knowledge information processing, where databases and knowledge-bases are frequently used as simulation or thinking experiment tools. From now on, we would like to introduce an example of legal reasoning. We issue the following queries:

- If we follow A's theory, what judgment will be predicted for a new case?
- If a legal precedent has fact B, how can we conclude that a new case is similar to the precedent and can expect a similar judgment?
- What information is necessary in the database to judge that a new case is innocent?

Such queries are generally non-standard, viz., subjunctive, hypothetical, and conditional [7]. This leads to the question: how can we represent them in partial information databases?

From a representation point of view, there are two major streams: value-based and identity-based [18]. In value-based representation, an object is represented by combination of values where lack of information is usually treated as a null value. Relational databases and deductive databases are typical examples of this. However, such representation is very inefficient for partial information because an object must be separated into segments such as binary relation sets, and additional constraints among segments are needed. On the other hand, in identity-based representation, an object consists of an object identifier and an arbitrary number of properties. It is more appropriate to represent partial information. Object-oriented databases are an example of this. However, as the semantics is not necessarily declarative, query processing for partial information has not been made clear. Recently, there have been many works on deductive object-oriented databases (DOOD) as the new database which integrate the features of value- and identity-bases representation [10, 6, 4, 23, 8, 9, 1, 3]. In the approach, query processing is defined declaratively. Because problems with partial information result from both representation and query processing, we adopt the DOOD approach.

*Information Science Dept., Mitsubishi Research Institute Inc. e-mail: {nishioka,tojo}@mri.co.jp

In our project, we have designed and developed a DOOD language called $\mathcal{QUIXOTE}$, which is a logic programming language, into which object-orientation concepts such as object identity, type hierarchy, property inheritance, and method are embedded by subsumption constraints [22, 12, 20, 21, 24]. From a representation point of view, value-based and identity-based representations are integrated into the language. From a viewpoint of reasoning of partial information, the language has features such as hypothetical reasoning and abduction. The module mechanism contributes to both objectives. Furthermore, there are some other features such as transaction and persistence. We have shown its effectiveness and efficiency on a variety of knowledge information processing applications [19, 14, 15, 16].

The contribution of this paper is to introduce the mechanism of query processing for partial information databases and shows its effectiveness as a tool for future various applications in artificial intelligence.

In Section 2, we show an example of legal reasoning to illustrate the sort of problems we are tackling. In Section 3, we outline specific features of $\mathcal{QUIXOTE}$, particularly representation of partial information. In Section 4, we explain query processing for a $\mathcal{QUIXOTE}$ database.

2 An Example from Legal Reasoning

In this section, to provide a concrete example of what we mean by partial information processing, we take the following new case related to "*karōshi*" (death from overwork):

> *Mary*, a *driver* employed by a 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 explanation of legal reasoning, and then secondly, show the kind of knowledge-base necessary for the example.

2.1 Legal Reasoning Process

Usually, the analytical legal reasoning process consists of three steps: *fact finding*, *statutory interpretation*, and *statutory application*. Among them, we focus on statutory applications, which can be considered:

analogy detection: Given a new case, similar precedents to the case are retrieved from existing precedents.

rule transformation: Precedents (interpretation rules) extracted by analogy detection are abstracted until the new case can be applied to them.

deductive reasoning: Apply the new case in a deductive manner to abstracted interpretation rules transformed by rule transformation.

In these three steps, the analogy detection strategy is essential in legal reasoning for *more efficient* detection of *better* precedents, which decides the quality of the results. To investigate the $\mathcal{QUIXOTE}$'s potential for legal reasoning, we developed an experimental system [19, 24]. Here we describe a simplified example of legal reasoning.

2.2 Example of Legal Knowledge

First, we consider legal knowledge. Assume that the *labor law* and the *theory* of the application of the law is already formulated as follows:

labor law: An organization is responsible to compensate its employee, if the judgment of the case is for "insurance."

theory: If the judgment of a case is both job-causality and job-execution at the same time, then the judgment of the case is for "insurance."

Further, assume that there are two precedents related to the law and already abstracted [1]:

precedent 1 (job-execution): If an employee Z in the status of Y causes X, then the judgment says that X is considered part of "job-execution."

precedent 2 (job-causality): If X occurs as the result of Z during an activity Y as a part of job, then the judgment says that X is considered "job-causal."

Note that these statements are abstracted from certain concrete precedents by rule-transformation, and the variables $X, Y, Z, \cdots$ are abstracted from concrete concepts which appeared in original precedents. We will show how they are abstracted in Section 4.4.

Next, we will divide the problem into two parts: knowledge representation and query processing. To represent the above knowledge, we must prepare the followings:

Representation of concepts: Concepts, such as "company," "break," and "heart-attack," should be defined.

Representation of relations between concepts: Relations between concepts, such as "Mary is a driver," "heart-attack is a kind of disease," and "Mary is employed by a company S" should be represented.

Representation of intensional objects: Intensional descriptions of objects, such as "if $\sim$, then $\cdots$," which appear in statutes in the above example, should be treated.

Classification of knowledge: Knowledge with properties which differ depending on the situation should be managed.

In Section 3, we will discuss how this knowledge is effectively represented in $\mathcal{QUIXOTE}$.

Finally, consider queries and answers for the above database.

query 1: According to the past precedents, what kind of judgment can we predict for the new case?

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

Considering partiality, the following features are required:

[1] In this paper, we omit the rule transformation step and assume abstract interpretation rules are given.

Selection of resources: The part "according to the ~" in the above queries corresponds to selection of resources for query processing, because such knowledge is stored on the database.

Lacking information: As both information in a query and a database might be partial, any information which is lacking should be provided. This corresponds to hypotheses in a query and assumption in the database.

In Section 4, we will discuss how *QUIXOTE* query processing effectively realizes these facilities.

3 Knowledge Representation in *QUIXOTE*

In this section, we overview the knowledge representation features of *QUIXOTE*.

3.1 Object Terms and Subsumption Constraints

Simple concepts can be represented as *basic objects*. We assume a set B of basic objects. For example, the following are basic objects:

$$mary, driver, employee, male, female, person$$

The set of basic objects are partially ordered by $\preceq$. For example.

$$mary \preceq driver,\ driver \preceq employee,$$
$$male \preceq person, female \preceq person$$

Relations between concepts such as "Mary is a driver," and "heart-attack is a kind of disease" can be represented by this partial ordering. For simplicity, we assume that $\preceq$ is strict order without circularity.

We represent complex concepts, such as a "company whose name is S," by *object terms*:

$$company[name = s],$$

where *company* is a basic object, *name* is a label, and s is an object term as the value of *name*. There are two kinds of labels: $l \in L_i$ takes a single value and $l^* \in L_s$ takes a set value, where $L_i \cap L_s = \emptyset$. l is called a single value label and l^* is a set value label. L_i is a subset of B. Similarly, there are two kinds of variables: $X \in V_i$ (a single value variable) and $X^* \in V_s$ (a set value variable).

An *object term* is defined as follows:

Definition 1 Object Term
Let $o \subset B$, $l_1, \cdots, l_n \in L_i$ where l_i and l_j $(i \neq j)$ are different, and $t_1, \cdots, t_n$ be object terms or variables $(\in V_i)$, then

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

is an *object term*. When $n = 0$, we simply write o instead of $o[\,]$. □

For example, *apple* and *apple*[*color* = *green*] are object terms. An object term with variables is called a *parametric* object term.

Partial order $\preceq$ among basic objects is extended to *subsumption relation* $\sqsubseteq$ among object terms as follows:

Definition 2 Subsumption Relation
Given two object terms without variables, $o[l_1 = t_1, \cdots, l_n = t_n]$ and $o'[l'_1 = t'_1, \cdots, l'_m = t'_m]$,
if $o \preceq o'$ and $\forall l'_j, \exists l_i \quad l_i = l'_j \wedge t_i \sqsubseteq t'_j$,
then $o[l_1 = t_1, \cdots, l_n = t_n] \sqsubseteq o'[l'_1 = t'_1, \cdots, l'_m = t'_m]$,
where $1 \leq j \leq m$ and $1 \leq i \leq n$. □

Example 1 Subsumption Relations

$$apple[color = green] \sqsubseteq apple,$$
$$male[age = 30, occupation = guitarist] \sqsubseteq person[occupation = musician],$$

where $male \sqsubseteq person$ and $guitarist \sqsubseteq musician$. □

In the case of object terms with variables, usually a co-reference relation is considered in the definition. For example, $o[l_1 = X, l_2 = X] \sqsubseteq o[l_1 = X, l_2 = Y]$. For details, see [20].

In a set of object terms, we assume that there is no common variables among elements. The subsumption relation among sets of object terms is also defined. Given two sets of object terms, $\{o_1, \cdots, o_n\}$ $(= S_1)$ and $\{o'_1, \cdots, o'_m\}$ $(= S_2)$, subsumption relation $\sqsubseteq_H$ among sets is defined in Hoare order as follows:

$$S_1 \sqsubseteq_H S_2 \overset{def}{=} \forall o_i \in S_1, \exists o'_j \in S_2\ o_i \sqsubseteq o'_j$$

Although the Hoare order is not partial, the representative of an equivalence class can be easily defined as a set where any two elements cannot be ordered, and the set of representatives is partially ordered. So we assume without loss of generality that $\sqsubseteq_H$ is a partial order.

Since lattice construction from a partially ordered set is a well known process, we assume that a set O of object terms (without variables) with $\top$ and $\bot$ is a lattice $(O, \sqsubseteq, \top, \bot)$ without loss of 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. Sets of object terms without variables constitute another lattice. Given two sets, S_1 and S_2, we can define meet and join operations ($\Downarrow$ and $\Uparrow$, respectively) under Hoare order as follows:

$$S_1 \Downarrow S_2 \overset{def}{=} \{e_1 \downarrow e_2 | \, e_1 \in S_1, e_2 \in S_2\}$$
$$S_1 \Uparrow S_2 \overset{def}{=} S_1 \cup S_2$$

where $\{\top\}$ is the top of the lattice and $\{\ \}$ is the bottom.

3.2 Partial Information as Subsumption Constraints

In the previous section, we showed how to represent so-called "is-a" and "a-kind-of" relations between concepts in *QUIXOTE*. To represent relations such as "Mary is employed by company S," we use a relation between an object term and a property of another object term:

$$mary.employer \cong company[name = s].$$

Given an object term, o, the value of a label l (l^*) of o is denoted by $o.l$ ($o.l^*$). An object term also may be used as a label: $o.o'$ denotes a value of o' of o, where o' is considered a single value label. $o.l$ ($o.l^*$) and $o.o'$ are called *dotted terms*. Thus, the term *mary.employer* can be read "Mary's employer." The subsumption constraints of an object term are defined by using dotted terms as follows:

Definition 3 Subsumption Constraint
Let t_1, t_2 be object terms, single value variables, or dotted terms with single value labels, then $t_1 \sqsubseteq t_2$ is a *subsumption constraint*. In the case of a set, if t_1^* and t_2^* are sets of object terms, set value variables, or dotted terms with set value labels, then $t_1^* \sqsubseteq_H t_2^*$ is also a *subsumption constraint*. If t_1 (t_1^*) or t_2 (t_2^*) includes an object term o, its related constraint is called a subsumption constraint *of* o. □

When $t_1 \sqsubseteq t_2 \wedge t_1 \sqsupseteq t_2$ ($t_1^* \sqsubseteq_H t_2^* \wedge t_1^* \sqsupseteq_H t_2^*$), we denote $t_1 \cong t_2$ ($t_1^* \cong_H t_2^*$) [2].

A set of subsumption constraints is saturated and reduced by applying the following rewriting rules:

$$\begin{array}{rcl} x \sqsupseteq y & \Rightarrow & y \sqsubseteq x \\ x \sqsubseteq y,\ y \sqsubseteq z & \Rightarrow & x \sqsubseteq z \\ x \sqsubseteq y,\ x \sqsubseteq z & \Rightarrow & x \sqsubseteq (y \downarrow z) \\ y \sqsubseteq x,\ z \sqsubseteq x & \Rightarrow & (y \uparrow z) \sqsubseteq x \\ x \cong y,\ y \cong z & \Rightarrow & x \cong z \\ x \sqsubseteq y,\ y \sqsubseteq x & \Rightarrow & x \cong y \\ o[\cdots, l{=}x, \cdots] \sqsubseteq o'[\cdots, l{=}y, \cdots],\ o \sqsubseteq o' & & \\ & \Rightarrow & x \sqsubseteq y \end{array}$$

where $x \sqsubseteq x$ and $x \cong x$ are removed in the procedure. The termination and confluency of the above rules are proved in [13]. Similar rules are also defined for set constraints.

Consider an object term with subsumption constraints (called *attribute term*), $o|C$, where o is an object term, and C is a set of subsumption constraints. An *intrinsic property* is a pair of a label and a value in o. An *extrinsic property* is a subsumption constraint of o in C of an attribute term. If both an intrinsic property and an extrinsic property have the same label, then the extrinsic property is removed. That is,

if $o[\cdots, l{=}t_1, \cdots]|\{o.l\ op\ t_2\} \cup C$,
then $o.l\ op\ t_2$ is removed,

where op is $\sqsubseteq, \sqsupseteq$, *or* $\cong$. That is, $o[\cdots, l{=}t, \cdots]|\{o.l = t\} \cup C$ is assumed if $o[\cdots, l{=}t, \cdots]|\{o.l\ op\ t'\} \cup C$ where C does not contain a subsumption constraint for $o.l$. When a label l does not appear, neither in an intrinsic property nor in an extrinsic property in an attribute term, $o|C$, $\bot \sqsubseteq o.l \sqsubseteq \top$ is assumed. We also use the following syntax sugars:

$$\begin{array}{rcl} o|\{o.l \sqsubseteq t\} \cup C & \Longleftrightarrow & o/[l \rightarrow t]|C \\ o|\{o.l \sqsupseteq t\} \cup C & \Longleftrightarrow & o/[l \leftarrow t]|C \\ o|\{o.l^* \sqsubseteq_H s\} \cup C & \Longleftrightarrow & o/[l^* \rightarrow_H s]|C \\ o|\{o.l^* \sqsupseteq_H s\} \cup C & \Longleftrightarrow & o/[l^* \leftarrow_H s]|C \\ o|\{o.l \cong t\} \cup C & \Longleftrightarrow & o/[l = t]|C \\ o|\{o.l^* \cong_H s\} \cup C & \Longleftrightarrow & o/[l^* =_H s]|C \end{array}$$

For *property inheritance*, only extrinsic properties are inherited according to the subsumption relation among object terms as follows:

Definition 4 Property Inheritance
If $o \sqsubseteq o'$ and o' does not have an intrinsic property of a label l, then $o.l \sqsubseteq o'.l$ and $o.l^* \sqsubseteq o'.l^*$. □

That is, by applying of a label, which is not included in an intrinsic property of the object terms, the subsumption relation between object terms makes its property inheritance monotonic. Property inheritance *exception* corresponds to the above restriction of extrinsic properties.

Example 2 Property Inheritance

1) If $apple/[color = red]$,
then $apple[weight = heavy]/[color \rightarrow red]$, but $apple[color = green]$ does not inherit $color \rightarrow red$.

2) If $apple[weight = heavy]/[area^* \leftarrow_H \{aomori\}]$ and $apple[color = green]/[area^* \leftarrow_H \{nagano\}]$,
then $apple/[area^* \leftarrow_H \{aomori, nagano\}]$
(by the join operation between sets). □

Note that, in the cases of $\leftarrow$ and $\leftarrow_H$, extrinsic properties are inherited upward by the above rule, while intrinsic properties are not, even though $apple[color = green]$ is $apple[color = green]/[color = green]$.

As property inheritance is constraint inheritance in $\mathcal{QUIXOTE}$, *multiple inheritance* corresponds to the merging of constraints without preferences.

3.3 Intensional Objects by Rules

An object in $\mathcal{QUIXOTE}$ consists of an object term and a set of methods. An object term without variables plays the role of an object identifier (oid), while each extrinsic property plays the role of a method. That is, a label (or an object term used as a label) corresponds to a message and the value corresponds to the return value.

Such an object can be defined *intensionally* in the form of a rule.

Definition 5 Rule
Let a_0 ($=o_0|C_0$), a_1 ($=o_1|C_1$), $\cdots$, a_n ($=o_n|C_n$) be attribute terms and D be a set of subsumption constraints, then

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

is a *rule*, where C_0 may not contain any subsumption relation between object terms. a_0 is called the *head* and $a_1, \cdots, a_n \mid\mid D$ is called the *body*. □

The rule can be transformed into

$$o_0|C_0 \Leftarrow o_1, \cdots, o_n \mid\mid C_1 \cup \cdots \cup C_n \cup D,$$

where C_0 is called a *head constraint* and $C_1 \cup \cdots C_n \cup D$ is called a *body constraint*. Further, a body constraint can be divided into a set A of constraints containing dotted terms and a set C of other constraints, where A and C are disjoint. The restriction of C_0 in the above definition is to avoid destruction of the lattice by assertion of a subsumption relation during derivation. If a body is empty, it is called a *fact*. Intuitively, a

[2] Although we do not describe the semantics of $\mathcal{QUIXOTE}$, they may be outlined in three parts:

1) An object term is mapped into a *labeled graph* as a subclass of a hyperset.
2) The subsumption relation among object terms corresponds to a *bisimulation relation* among labeled graphs.
3) A label or an object term used as a label corresponds to a *function* on a set of labeled graphs. Here the subsumption relation among labels is not considered.

For details, see [20]

rule means that if a body is satisfied then a head is satisfied.

For the case in which there is no head constraint, $\mathcal{QUIXOTE}$ may be considered an instance of CLP(X) [11], where a constraint domain is a set of labeled graphs — as a subclass of hypersets and (extended) subsumption relations — and constraints set A is ignored in the procedural semantics. Without set subsumption constraints, $\mathcal{QUIXOTE}$ becomes a subclass of CLP(AFA), with a hyperset constraint domain [13]. The head constraint makes $\mathcal{QUIXOTE}$ different from an instance of CLP(X).

Generation of oids during derivation and property inheritance make the procedural semantics of $\mathcal{QUIXOTE}$ unique and complex, that is, synchronous merge operations are needed in 'OR parallel'.

Example 3

1) If $apple/[taste \rightarrow sour] \wedge fruit/[taste \rightarrow sweet]$ and $apple \sqsubseteq fruit$, then $apple/[taste \rightarrow sweet \downarrow sour]$.
2) If $lottery[num = X]/[prize \rightarrow Y] \Leftarrow B_1 \wedge lottery[num = X]/[prize \rightarrow Y] \Leftarrow B_2$, then the possibility of merging the two properties must be checked after evaluation of both rules.
3) If $dog[body = small]/[bark \rightarrow noisy] \wedge dog[body = X]/[bark \rightarrow Y] \Leftarrow B$, then the subsumption relation between $dog[body = small]$ and $dog[body = X]$ and the related property inheritance are not decided until X is instantiated.

□

3.4 Modules

In $\mathcal{QUIXOTE}$, a set of rules can be modularized:

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

where m is a *module identifier* (mid) (in the form of an object term) and $r_1, \cdots, r_n$ are rules. For simplicity, we use the notation a module m instead of a module with a mid m. Modules can be nested. When a mid has variables, it is called a *parametric module*. Variables in a mid are global in the module, that is, variables in a mid can be shared by rules in the module. A module can be explicitly referred to by rules in other modules. We extend the definition of a rule as follows:

Definition 6 Rules
Let $m_0, m_1, \cdots, m_n$ be mids, $a_0, a_1, \cdots, a_n$ attribute terms, and D a set of subsumption constraints. A rule is defined as follows:

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

□

The rule in the above definition means that a module m_0 has a rule such that if a_1 is satisfied in a module m_1, $\cdots$, and a_n is satisfied in a module m_n, then a_0 is satisfied in a module m_0. The rule may be transformed:

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

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

We introduce the module concept for the following objectives:

- modularization and classification of knowledge,
- co-existence or localization of inconsistent knowledge,
- temporal storage of tentative knowledge, and
- introduction of a modular programming style.

To meet these objectives, we define *submodule relation* among modules:

Definition 7 Submodule Relations
Given two modules, m_1 and m_2, a submodule relation $m_1 \sqsupseteq_S m_2$ means that m_1 inherits all the rules in m_2, when m_2 is called a *submodule* of m_1. □

The submodule relation specifies *rule inheritance*, while the subsumption relation specifies property inheritance. For exception, locality, and overriding of rule inheritance, see [22].

In the current implementation of $\mathcal{QUIXOTE}$, the names of object terms, the subsumption relation, and the submodule relation are global in a database, while the existence of objects and extrinsic properties are local. That is, if there is no (transitive) submodule relation between two modules, then their extrinsic properties do not mutually interfere. If there is no relation between two modules, inconsistent knowledge can coexist separately in them. Furthermore, answers to the same query by different modules may be different.

3.5 Database

We define a *database* or a *program* as a triple (S, M, R), where S, M, R correspond to definitions of subsumption relations [3] , submodule relations, and rules. Definitions of rules can be considered definitions of objects or definitions of modules.

An object term corresponds to value-based representation because each property in it is intrinsic. An attribute term, however, corresponds to identity-based representation because its oid does not change, even if extrinsic properties do. In this sense, a $\mathcal{QUIXOTE}$ object has the features of both the representations described in Section 1.

4 Query Processing

In this section, we will show how $\mathcal{QUIXOTE}$ can effectively process queries on partial information. In Section 4.1 we give further consideration about partiality of information, and show that hypothetical reasoning and abduction are useful techniques for processing partial information. In sections 4.2 and 4.3, we show how these techniques are realized in $\mathcal{QUIXOTE}$. Finally, in Section 4.4, we show how $\mathcal{QUIXOTE}$ can process the example given in Section 2.

4.1 What is Partial?

There are many points to be considered in a partial information database. First, we must recognize that database information may be partial or incomplete:

1) Necessary definitions might be lacking for any element of (S, M, R).
2) Facts or rules might be ambiguous or indefinite. Mutually inconsistent hypotheses may occur.

[3] Only the $\preceq$-relation is defined in S.

3) A $\mathcal{QUIXOTE}$ object might be incompletely defined: indefinite numbers of extrinsic properties may be present, each of which may be specified in the form of a constraint.

For example, consider the analogy among legal precedents (with incomplete descriptions) is indispensable in legal reasoning and hidden knowledge (undescribed knowledge) is important in natural language processing. In such areas, complete description cannot be expected.

A second point that must be considered, is the special query processing requirements when dealing with this kind of partial information database:

1) What will be returned for this query if a user inserts some information (candidates for lacking information) as hypotheses?
2) What information is lacking in this database for successfully answering this query?
3) When mutually inconsistent data or knowledge is stored separately in different modules in a database, which is better for this application?
4) When hypotheses generated during query processing are used by successive queries, how are they controlled and where are they stored?
5) Why is this answer (the constraint) returned as an answer for this query? What knowledge is used for the answer?

A query in $\mathcal{QUIXOTE}$ is in the form of

query if *hypotheses*

where *hypotheses* corresponds to 1). Such hypotheses may be incrementally inserted into the database for 4). A *query* may be issued to different modules for 3). An answer in $\mathcal{QUIXOTE}$ is in the form of

if *assumptions* then *answer* because *explanation*

where *assumptions* corresponds to 2), *explanation* corresponds to 5), and, as in CLP, *answer* is a set of subsumption constraints. These two kinds queries are related as follows:

query$_1$: ?-*query*.
answer$_1$: if *assum* then *answer*.
query$_2$: ?-*query* if *assum*.
% *assum* in **query**$_1$ is used as hypotheses of **query**$_2$.
answer$_2$: (unconditionally) *answer*.
% The same answer is returned without *assum*.

4.2 Hypothetical Reasoning

In general, hypothetical reasoning in a database DB is defined as reasoning in a database $DB \cup H$, where H is a set of hypotheses [2].

Example 4 Path Relation
Consider the following database DB:

$arc[from=a, to=b];;$
$arc[from=c, to=d];;$
$path[from=X, to=Y] \Leftarrow arc[from=X, to=Y];;$
$path[from=X, to=Y] \Leftarrow arc[from=X, to=Z], path[from=Z, to=Y].$

where ";;" is a delimiter between rules. For a query ?-$path[from=a, to=d]$, the answer is simply **no**, while, for the same query under a hypothesis such that DB has a fact $arc[from=b, to=c]$, the answer is **yes**. □

Just as a $\mathcal{QUIXOTE}$ database is defined as (S, M, R), a hypothesis also consists of a triple (H_S, H_M, H_R), where H_S, H_M, and H_R are a set of hypotheses for S, M, and R. A query Q with a hypothesis (H_S, H_M, H_R) to a database (S, M, R) is equivalent to a query Q without hypotheses to a database $(S \cup H_S, M \cup H_M, R \cup H_R)$. Hypothesis insertion is done before processing query Q. Even if such insertion requires reconstruction of the lattice of object terms and the submodule graph of modules, there are no problems logically. The only possible penalty is in terms of performance efficiency.

Example 5 Cider Example
Consider the following database on *cider*:

$usa \sqsupseteq_S west;;uk \sqsupseteq_S west;;$
$japan : \{cider/[source=soda\_pop]\};;$
$west : \{cider/[source=apple, process=ferment]\};;$
$uk : \{cider/[alcohol=yes]\};;$
$usa : \{cider/[alcohol=no]\}.$

For a query ?-$japan : cider/[source=X, alcohol=Y]$, the answer is that $X = soda\_pop$ and Y is unbound ($\bot \sqsubseteq Y \sqsubseteq \top$). For a query ?-$japan : cider/[source=X, alcohol=Y]$ if $japan \sqsupseteq_S usa$, the answer is that X=**inconsistency** and $Y=no$. □

Note that $cider/[alcohol=yes]$ and $cider/[alcohol=no]$ are inconsistent but they are stored in different modules (*uk* and *usa*), neither of which is a submodule of the other. *west* is common knowledge of *uk* and *usa*.

In the sequence of queries, such hypotheses are incrementally inserted into a database. To control such insertions, *nested transaction* is introduced into $\mathcal{QUIXOTE}$: that is, even if a database is reorganized by hypotheses, the original image is recovered by *rollback* operations.

Example 6 Query Sequence
The following is an example of a query sequence:

?-open_db(DB).	% Open a database named DB.
?-begin_trans.	% Begin a transaction (level 1).
?-q_1;;H_1.	% Same as ?-q_1 to $DB \cup H_1$.
?-begin_trans.	% Begin a transaction (level 2).
?-q_2;;H_2.	% Same as ?-q_2 to $DB \cup H_1 \cup H_2$.
	% Hypotheses are incrementally inserted.
?-abort_trans.	% Abort a transaction (level 2).
	% H_2 is rolled back.
?-q_3;;H_3.	% Same as ?-q_3 to $DB \cup H_1 \cup H_3$.
?-end_trans.	% Commit a transaction (level 1).
	% DB is updated to $DB \cup H_1 \cup H_3$.
?-close_db(DB).	% Close a database DB.

where ";;" is a delimiter between a query and a hypothesis. □

4.3 Abduction of Subsumption Constraints

First, consider a simple database consisting of a rule and a fact:

$john/[age = A] \Leftarrow john/[twin\_brother = paul],$
$\qquad paul/[age = A]$[4]$;;$
$paul/[age = 30].$

For a query $?\text{-}john/[age = X]$, what answer is expected? Although *paul*'s age is specified, there is no fact stating that *john* and *paul* are *twin_brother*. Without making any assumptions, the query fails. However, as we focus on the partiality of the information, the lack of information suggests an assumption be taken. So, in $\mathcal{QUIXOTE}$, the answer is that if $john.twin\_brother \cong paul$ then X is 30, that is, unsatisfied constraints of other objects' extrinsic properties in bodies are assumed.

In logic programming, finding a lack of information or unsatisfiable subgoals corresponds to *abduction*, that is, hypothesis or explanation generation [5]. Remember that a rule in $\mathcal{QUIXOTE}$ can be represented as follows:

$$o_0|C_0 \Leftarrow o_1, \cdots, o_n \;||\; A \cup C,$$

where object terms, $o_1, \cdots, o_n$, are considered existence checks of corresponding objects, C_0, a set of dotted constraints of o_0, is considered assertional constraints, and C, a set of variable constraints, is considered a set of constraints to be satisfied. A is considered constraints of other objects' extrinsic properties. In $\mathcal{QUIXOTE}$, only A is taken as assumption, that is, even if body constraints about dotted terms are not satisfied, they are taken as a conditional part of an answer. Although A and C are disjoint, when variables in C are bound by dotted terms during query processing, constraints with the variables in C are moved into A. If the subsumption relation between object terms is taken as assumption, it might destroy the soundness of the derivation because it affects property inheritance and does not guarantee results in the former derivation.

Abduction is closely related to procedural semantics. Here we will only briefly explain the relation. In general, derivation by query processing in CLP is the finite sequence of a pair (G, C) of a set G of goals and a set C of constraints:

$(G_0, C_0) \Rightarrow (G_1, C_1) \Rightarrow \cdots \Rightarrow (G_{n-1}, C_{n-1}) \Rightarrow (\emptyset, C_n).$

On the other hand, derivation in $\mathcal{QUIXOTE}$ is a finite directed acyclic graph of the triple (G, A, C) of the set G of goals, the set A of assumptions, and the set C of constraints. Remember that a rule is in the following form:

$$o_0|C_0 \Leftarrow o_1, \cdots, o_n \;||\; A \cup C$$

A query is also transformed in the form of $?\text{-}o_1, \cdots, o_n \;||\; A_0 \cup C_0$, i.e., a triple $(\{o_1, \cdots, o_n\}, A_0, C_0)$. For a node $(\{G\} \cup G_i, A_i, C_i)$, a rule $G'|C' \Leftarrow B \;||\; A \cup C$, and $\exists\theta\; G\theta = G'\theta$, where B is a set of object terms and θ is a substitution, the transformed node is:

$$((G_i \cup B)\theta,\; (A_i\theta \setminus C'\theta) \cup A\theta,\; (C_i \cup C \cup C')\theta).$$[5]

The derivation image is illusted as in Figure 1. If there are two nodes, (G, A, C) and (G, A', C), where $A \subseteq A'$, then the derivation path of (G, A', C) is thrown away, i.e., only the minimal assumption is made. If there are two nodes, (G, A, C) and (G, A, C'), then they are merged into $(G, A, C \cup C')$.

Consider the following example of two speakers with different knowledge:

Example 7 Quine's (modified) example
There are two speakers, a and b:

- a: If Bizet and Verdi are compatriots, then Bizet is Italian.
- b: If Bizet and Verdi are compatriots, then Verdi is French.

Each of the speakers has different hidden knowledge, that is, a assumes that Verdi is Italian. Such hidden knowledge is treated as an assumption in $\mathcal{QUIXOTE}$ query processing. The example is written in $\mathcal{QUIXOTE}$ as follows:

$nation \sqsupseteq italy;; nation \sqsupseteq france;;$
$speaker\_a \sqsupseteq_S common;; speaker\_b \sqsupseteq_S common;;$
$speaker\_a : \{$
$\quad bizet/[nationality = italy] \Leftarrow$
$\qquad compatriots[per1 = bizet, per2 = verdi]\};;$
$speaker\_b : \{$
$\quad verdi/[nationality = france] \Leftarrow$
$\qquad compatriots[per1 = bizet, per2 = verdi]\};;$
$common : \{$
$\quad compatriots[per1 = X, per2 = Y] \Leftarrow$
$\qquad X/[nationality = N1], Y/[nationality = N2]$
$\qquad || \{N1 \sqsubseteq nation, N2 \sqsubseteq nation, N1 \cong N2\};;$
$\quad bizet;; verdi\},$

For a query $?\text{-}speaker\_a : bizet/[nationality = X]$, the answer is that
if $bizet.nationality = verdi.nationality$
and $verdi.nationality = italy$
then $X = italy$.

On the other hand, for a query $?\text{-}speaker\_b : verdi/[nationality = X]$, the answer is that
if $verdi.nationality = bizet.nationality$
and $bizet.nationality = france$
then $X = france$. □

Even if there is insufficient data in a database, $\mathcal{QUIXOTE}$ explicates hidden knowledge.

The query processing process is more complex than conventional processing because modes may be merged. Additionally, we might get different answers from the same query if the query is made to different modules. In such cases, the derivation process of an answer is returned including an *explanation*, if necessary. By referring to this explanation, users can verify which rules are used for the answer.

[4] It is a simplified version of a rule $X/[age = A] \Leftarrow X/[twin\_brother = Y], Y/[age = A]$.

[5] Note that, here, we ignore that some elements in $(C_i \cup C \cup C')\theta$ might be moved into $(A_i\theta \setminus C'\theta) \cup A\theta$.

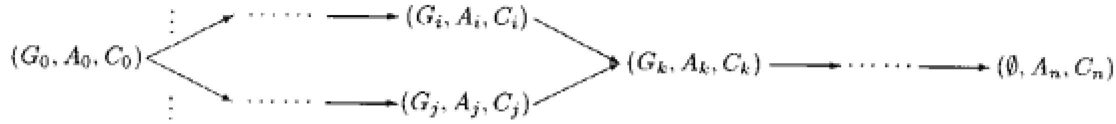

Figure 1: Derivation Network

4.4 Legal Reasoning

Now we are ready to show how *QUIXOTE* can effectively represent knowledge and process queries for the legal reasoning example given in Section 2.

The new case in Section 2 is represented as a module *new-case* in *QUIXOTE* as follows:

$new\text{-}case : \{$
$\quad new\text{-}case/[who = mary, while = break,$
$\qquad result = heart\text{-}attack];;$
$\quad relation[state = employ, employee = mary]$
$\qquad /[affiliation = organization[name = \text{“}S\text{”}],$
$\qquad job \rightarrow driver]\}$

The labor law and the theory in Section 2 are represented as the following *QUIXOTE* *labor-law* and *theory* modules:

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

Two abstracted precedents in Section 2 are represented as follows:

$case_1 : \{$
$\quad judge[case = X]/[judge \rightarrow job\text{-}execution]$
$\qquad \Leftarrow relation[state = Y, employee = Z]/[cause = X]$
$\qquad ||\{X \sqsubseteq \mathbf{parm}.case, Y \sqsubseteq \mathbf{parm}.status,$
$\qquad Z \sqsubseteq \mathbf{parm}.employee\}\}.$
$case_2 : \{$
$\quad judge[case = X]/[judge \rightarrow job\text{-}causality]$
$\qquad \Leftarrow X/[while = Y, result = Z],$
$\qquad ||\{Y \sqsubseteq job, X \sqsubseteq \mathbf{parm}.case,$
$\qquad Y \sqsubseteq \mathbf{parm}.while, Z \sqsubseteq \mathbf{parm}.result\}\}.$

Note that variables X, Y, and Z in both rules are restricted by the properties of the object **parm**. That is, **parm** controls the abstraction level (the range of variables). Such precedents are retrieved from the precedent module by analogy detection and are abstracted by rule transformation.

We define the **parm** object as follows:

$parm : \{\mathbf{parm}/[case = case, state = relation, while = job,$
$\qquad result = disease, employee = person]\}.$

This object is a result of abstraction of precedents and is used for control of predicting judgments.

To use parm for $case_1$ and $case_2$, we define the following submodule relation:

$$parm \sqsupseteq_S case_1 \cup case_2.$$

It is dynamically defined during rule transformation, because the choice of precedents is experimental.

Furthermore, we define the subsumption relations:

$$\begin{array}{rcl} case & \sqsupseteq & new\text{-}case \\ relation & \sqsupseteq & employ \\ disease & \sqsupseteq & heart\text{-}attack \\ job & \sqsupseteq & break \\ person & \sqsupseteq & mary \\ job\text{-}causality & \sqsupseteq & insurance \\ job\text{-}execution & \sqsupseteq & insurance \end{array}$$

Such relations are defined in advance by the definition of subsumption relations.

Then, we can make the following queries to generate a hypothesis from the above database:

1) *query 1*: According to the past precedents, what kind of judgment can we predict for the new case?
2) *query 2*: According to the labor law, what kind of responsibility should the organization which Mary is affiliated to have?

They are represented and processed as follows:

1) If *new-case* inherits *parm* and *theory*, then what kind of *judgment* can we predict?

   $?\text{-}new\text{-}case : judge[case = new\text{-}case]/[judge = X];;$
   $\quad new\text{-}case \sqsupseteq_S parm \cup theory.$

   We can get three answers:

   - $X = job\text{-}causality$
   - if $new\text{-}case : judge[case = new\text{-}case]$ has a property $judge \sqsubseteq job\text{-}execution$, then $X \sqsubseteq insurance$.
   - if $new\text{-}case : relation[state = employ, employee = mary]$ has a property $cause = new\text{-}case$, then $X \sqsubseteq insurance$.

The first answer is returned unconditionally, while the last two include assumptions.

2) If *new-case* inherits *labor-law* and *parm*, then what kind of responsibility should the organization which Mary is affiliated to have?

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

   We can get two answers:

   - if *new-case* : *judge*[*case* = *new-case*] has a property *judge* $\sqsubseteq$ *job-execution*, then $X \sqsubseteq$ *compensation*[*object* = *mary*, *money* = *salary*]
   - if *new-case* : *relation*[*state* = *employ*, *employee* = *mary*] has a property *cause* = *new-case*, then $X \sqsubseteq$ *compensation*[*object* = *mary*, *money* = *salary*]

Both these answers include assumptions. That is, if hypotheses are not generated, no answers are returned.

For analogy detection, the **parm** object plays an essential role in determining how to abstract rules, as in $case_1$ and $case_2$, what properties are to be abstracted in parm, and what values are to be set as **parm** properties. In this experimental system, which has additional functions different from *QUIXOTE*, we have experimented with not only hypothetical reasoning and abduction, but also such abstraction, that is, analogy detection.

5 Concluding Remarks

In this paper, we introduced a tool for coping with partial information in databases, that is, the query processing in *QUIXOTE*. The points we claimed are summarized as follows:

- Representation and inference of partial information is essential in knowledge information processing.
- Non-standard queries and answers which include hypothetical reasoning and abduction are important for thinking experiments in partial information databases.
- *QUIXOTE*, a DOOD language based on subsumption constraints, provides such query processing facilities, which have shown their usefulness in applications such as legal reasoning.

As database application fields diverse, we have confronted the need of partial and incomplete information processing, especially in scientific databases, natural language databases, and knowledge-bases. Although traditional databases apply the most simple and clearly defined aspects of the real world, we have to cope with other complex and 'gray' aspects in more complex applications. There are not always well-defined borders between areas such as databases, programming languages, and artificial intelligence. As *QUIXOTE* was designed to meet the requirements of various applications in knowledge information processing, it has many features: it is more than just a deductive object-oriented database language, it is also a language for knowledge representation, situated programming, database programming, and constraint logic programming. To examine the database features alone in this paper, we could not explain all of the features — such as updating, transactions, persistence, and architectural characteristics — because of space limitation. We plan to produce further references on these details. In the meaning, please refer to [22, 21, 24].

We started to design the *QUIXOTE* language in 1990 and we have implemented several versions of the system. *QUIXOTE*, which can work under UNIX environments, has been released as ICOT free software for developing many knowledge information processing applications. We plan to extend the language further for heterogeneous, distributed, cooperative knowledge-base, and problem solving environments.

References

[1] S. Abiteboul and P. Kanellakis, "Object Identity as a Query Language Primitive," *Proc. ACM SIGMOD International Conference on Management of Data*, Portland, June, 1989.

[2] N. Cercone and G. McCalla (eds.), *The Knowledge Frontier — Essays in the Representation of Knowledge*, Springer-Verlag, 1987.

[3] S. Ceri and R. Manthey, "Overview of CHIMERA — The Conceptual Interface of IDEA," *S. Ceri's Lecture at ICOT*, Jan. 12, 1993.

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

[5] E. Charniak and D. McDermott, *Introduction to Artificial Intelligence*, Addison-Wesley, 1985.

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

[7] R. Demolombe (ed.), *Proc. Workshop on Non-standard Queries and Answers*, 2 vols, Toulouse, France, July 1-3, 1991.

[8] M. Kifer and G. Lausen, "F-Logic — A Higher Order Language for Reasoning about Objects, Inheritance, and Schema," *Proc. ACM SIGMOD International Conference on Management of Data*, pp.134-146, Portland, June, 1989.

[9] M. Kifer and J. Wu, "A Logic for Object-Oriented Logic Programming (Maier's O-Logic: Revisited)," *Proc. 8th ACM Symposium on Principles of Database Systems*, Philadelphia, Mar., 1989, pp. 379-393.

[10] W. Kim, J.-M. Nicolas, and S. Nishio (eds.), *Deductive and Object-Oriented Databases, (Proc. the First International Conference on Deductive and Object-Oriented Databases (DOOD89))*, North-Holland, 1990.

[11] J. Jaffar and J.-L Lassez, "Constraint Logic Programming," *Proc. 4th IEEE. Symposium on Logic Programming*, 1987

[12] Yukihiro Morita, Hiromi Haniuda, and K. Yokota, "Object Identity in *QUIXOTE*," *Proc. SIGDBS and SIGAI of IPSJ*, Oct., 1990.

[13] K. Mukai, "CLP(AFA): Coinductive Semantics of Horn Clauses with Compact Constraint," *Proc. the Second Conference on Situation Theory and Its Applications*, Kinloch Rannoch, Scotland, Sep., 1990.

[14] H. Tanaka, "Protein Function Database as a Deductive and Object-Oriented Database," *Proc. Second International Conference on Database and Expert System Applications*, Berlin, Apr., 1991.

[15] S. Tojo and H. Yasukawa, "Situated Inference of Temporal Information," *Proc. International Conference on Fifth Generation Computer Systems*, ICOT, Tokyo, June 1-5, 1992.

[16] S. Tojo, H. Tsuda, H. Yasukawa, K. Yokota, and Y. Morita, "*QUIXOTE* as a Tool for Natural Language Processing," *Proc. the Fifth International Conference on Tools with Artificial Intelligence*, Boston, Nov. 8-11, 1993.

[17] S. Uchida, R. Hasegawa, K. Yokota, T. Chikayama, K. Nitta, and A. Aiba, "Outline of the FGCS Follow-On Project," *New Generation Computing*, vol.11, pp.217-222, 1993.

[18] J.D. Ullman, "Database Theory — Past and Future," *Proc. the Sixth ACM Symposium on Principles of Database Systems*, 1987.

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

[20] H. Yasukawa and K. Yokota, "Labeled Graphs as Semantics of Objects," *Proc. SIGDBS and SIGAI of IPSJ*, Oct., 1990.

[21] H. Yasukawa, H. Tsuda and K. Yokota, "Object, Properties and Modules in *QUIXOTE*," *Proc. International Conference on Fifth Generation Computer Systems*, ICOT, Tokyo, June 1-5, 1992.

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

[23] K. Yokota and S. Nishio, "Towards Integration of Deductive Databases and Object-Oriented Databases — A Limited Survey," *Proc. Advanced Database System Symposium*, Kyoto, Dec., 1989.

[24] 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. International Conference on Fifth Generation Computer Systems*, ICOT, Tokyo, June 1-5, 1992.