# Extensions of Query Processing Facilities in Mediator Systems

Kazumasa Yokota
Okayama Prefectural University
yokota@c.oka-pu.ac.jp

Yutaka Banjou Takashi Kuroda
Kyoto University
{banjou,tkuroda}@kuis.kyoto-u.ac.jp

Takeo Kunishima
Nara Institute for Science and Technology
kunishi@is.aist-nara.ac.jp

CEUR-WS.org/Vol-8/paper-17.pdf

## Abstract

For advanced data-oriented applications in distributed environments, effective information is frequently obtained by integrating or fusing various autonomous information sources. There are many problems: how to resolve their heterogeneity, how to integrate target sources, how to represent information sources with a common protocol, and how to process queries. In this paper, we propose a new language, *QUIK*, as an extension of a deductive object-oriented database (DOOD) language, $\mathcal{QUIXOTE}$, and extend typical mediator systems. In this paper, we discuss various features of query processing facilities of QUIK: reducing inconsistency among information sources, identifying objects, searching alternative information sources by hypothesis generation, and applying multiagent-based strategies.

## 1 Introduction

Recently information sources in network environments are rapidly increasing not only in their quantity but also in their variety. Especially people can provide their own information easily through internet or intranet. We can get more effective information frequently by merging multiple sources, however, there are many problems for treating multiple sources as a virtual integrated database. Many search engines provide access facilities to multiple sources, however most of them are superficial text search but not content search.

In the Japanese FGCS project, we designed and implemented a deductive object-oriented database (DOOD) language or a knowledge representation language, $\mathcal{QUIXOTE}$ [YY92, YNTT94], and a heterogeneous, distributed, cooperative problem solving system, *Helios* [Yokota94, AY95]. However, to cope with integration of heterogeneous information sources, we have to redesign a new language for representing various information sources including DOOD, integrating them, and searching lacking information sources or alternative answers. In this paper, we discuss extensions of query processing facilities in mediator systems to integrate distributed information sources flexibly: an exchange model and mediator specification in QUIK, reducing inconsistency among information sources, identifying objects, searching alternative information sources by hypothesis generation, and applying multiagent-based strategies.

We discuss a mediator architecture and its extensions in Section 2 and explain a new language QUIK, as an extension of a DOOD language $\mathcal{QUIXOTE}$ in Section 3. We describe merging of subsumption relations in multiple information sources in Section 4 and extensions of query processing facilities in Section 5.

*The copyright of this paper belongs to the papers authors. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage.*

**Proceedings of the 4th KRDB Workshop**
**Athens, Greece, 30-August-1997**

(F. Baader, M.A. Jeusfeld, W. Nutt, eds.)

http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-8/

## 2 Mediator Architecture

Generally, a mediator system consists of a wrapper or an exchange model, which capsules an information source and provide a common protocol, and a mediator, which defined a view for related exchange models, as in [PAG96].

However, from a DOOD point of view, we can consider its various extensions as follows:

- Both an exchange model and a mediator should have powerful representation capability including DOOD.
- Integration of multiple information sources should maintain consistency of object-orientation features such as subsumption relation and property inheritance.
- Hypothetical reasoning such as conditional query processing and hypothesis generation as in $\mathcal{QUIXOTE}$ should be applied to distributed environments.

A mediator defines an integrated view for multiple information sources as in [PAG96]. In this paper, we consider the followings:

- As a mediator is a kind of information sources, we use a common language between a exchange model and mediator specification. That is, a mediator system can become hierarchical as in Figure 1. As we do not discuss transformation between an information source and a QUIK program in this paper, we use a term, *mediator*, generally without misunderstanding their differences.
- We introduce deductive object-oriented features, including object concepts, rules, and subsumption relation into the above language.
- We can define multiple exchange models for a single information source and parameterize their identifiers.
- We extend a concept of a mediator identifier or its name to search alternative sources, and provide a framework of introducing multiagent-based cooperation protocols and strategies in QUIK as in Helios.

As $\mathcal{QUIXOTE}$ does not have features for describing multiple information sources and merging subsumption relations, we design a new language called QUIK ($\mathcal{QUIXOTE}$ in Kyoto). By using QUIK, mediators are also treated as (virtual) information sources.

## 3 QUIK Program as a Mediator

An exchange model should be powerful and flexible for describing various information including semi-structured data. We propose a new unified language, QUIK, for describing an exchange model and mediator specification. Further, QUIK has features not only for describing complex objects, rules, and property inheritance, but also for generating hypotheses as alternative information sources.

### 3.1 QUIK Objects

A concepts of a QUIK object is taken from $\mathcal{QUIXOTE}$'s.

An object consists of an object identifier (oid) and a set of properties. The subsumption relation among oids makes property inheritance possible.

#### 3.1.1 Object Identity and Subsumption Constraints

An oid is in the form of a tuple called an *object term*. For example,

$$apple,$$

$$apple[color = red], \text{ and}$$

$$\begin{aligned}&cider[alcohol = yes,\\ &\quad product = process[source = apple,\\ &\qquad\qquad process = ferment]]\end{aligned}$$

are object terms, where *apple* is *basic*, but the latter two are *complex*. Although an infinite structure and set constructors are introduced, we do not explain them here for simplicity.

Given *subsumption relation* (partial order) $\sqsubseteq$ among basic object terms, the relation is extended among complex object terms as usual. For example,

$$apple \sqsupseteq apple[color = red].$$

Congruence relation $o_1 \cong o_2$ is defined as $o_1 \sqsubseteq o_2 \wedge o_1 \sqsupseteq o_2$. We assume that a set of object terms with the subsumption relation and special objects, $\top$ and $\bot$, constitutes a lattice without loss of generality because it is easy to construct a lattice from a partially ordered set, as in [ALN87]. Meet and join operations of object terms are denoted by $\downarrow$ and $\uparrow$, respectively.

*Properties* are defined as a set of subsumption constraints of an oid and used with the oid as follows:

$$\begin{aligned}&apple|\{apple.species \sqsubseteq rose,\\ &\quad apple.area \sqsupseteq_H \{aomori, nagano\}\},\end{aligned}$$

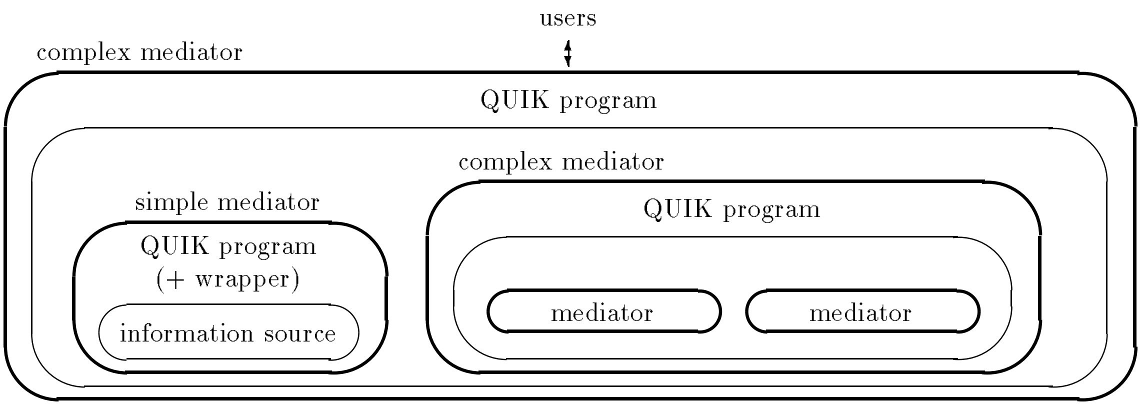

Figure 1: Hierarchical Structure of Mediators

where *apple* is an object term and the right hand side of | is a set of properties: $apple.species \sqsubseteq rose$ means that *apple*'s *species* is (subsumed by) *rose* and $apple.area \sqsupseteq_H \{aomori, nagano\}$ means that there are *aomori* and *nagano* in *apple*'s production *area*s. Here a relation between sets is defined as Hoare ordering based on subsumption relation:

$$S_1 \sqsubseteq_H S_2 \overset{\text{def}}{=} \forall e_1 \in S_1, \exists e_2 \in S_2,\ e_1 \sqsubseteq e_2.$$

Although Hoare ordering is not partial, we assume it as a partial order because the representative of an equivalence class modulo $\sqsubseteq_H$ is easily defined as a set where any element is not subsumed by other elements in the same set.

### 3.1.2 Property Inheritance

For *property inheritance*, we assume the following rule:

> if $o_1 \sqsubseteq o_2$, and neither $o_1$ nor $o_2$ has labels, $l$ and $l'$
> then $o_1.l \sqsubseteq o_2.l$ and $o_1.l' \sqsubseteq_H o_2.l'$,

where $o_1$ and $o_2$ are object terms, $l$ and $l'$ are labels, and $l$ and $l'$ take a single value and a set value, respectively. According to the rule, we get, for example,

> if $apple.species \sqsubseteq rose$,
> then $apple[color = red].species \sqsubseteq rose$,
> and
> if
> $apple[color = red].area \sqsupseteq_H \{fukushima\}$,
> then $apple.area \sqsupseteq_H \{fukushima\}$.

That is, $apple.species \sqsubseteq rose$ is downward inherited from *apple* to $apple[color = red]$, while $apple[color = red].area \sqsupseteq_S \{fukushima\}$ is upward inherited from $apple[color = red]$ to *apple*.

Note that there are two kinds of properties: properties in an object term and properties in the form of constraints. The former are called *intrinsic* and the latter are called *extrinsic*. Only extrinsic properties (subsumption constraints) are inherited according to the (extended) subsumption relation among object terms.

Intrinsic properties interrupt property inheritance as follows:

> Even if *apple* has $apple.color \cong green$, $apple[color = red]$ does not inherit $color \sqsubseteq green$
> because the intrinsic property $color = red$.

This corresponds to exception of property inheritance.

Multiple inheritance is defined as the merging of subsumption constraints. Such constraints are reduced as follows:

$$\begin{array}{lcl} p.l \sqsubseteq a \wedge o.l \sqsubseteq b & \Rightarrow & o.l \sqsubseteq a \downarrow b \\ a \sqsubseteq o.l \wedge b \sqsubseteq o.l & \Rightarrow & a \uparrow b \sqsubseteq o.l \\ o.l \sqsubseteq_H s_1 \wedge o.l \sqsubseteq_H s_2 & \Rightarrow & o.l \sqsubseteq_H s_1 \cup s_2 \\ s_1 \sqsubseteq_H o.l \wedge s_2 \sqsubseteq_H o.l & \Rightarrow & \{x \downarrow y \mid x \in s_1, y \in s_2\} \sqsubseteq_H o.l \end{array}$$

Note that the least upper bound of the two sets, $s_1$ and $s_2$, is defined as $s_1 \cup s_2$ under Hoare ordering, because $s_1 \cup s_2 \sqsubseteq_H \{e_1 \uparrow e_2 |\ e_1 \in s_1, e_2 \in s_2\}$. In the above example, the merging of $apple.area \sqsupseteq_H \{aomori, nagano\}$ and $apple.area \sqsupseteq_H \{fukushima\}$ is reduced to $apple.area \sqsupseteq_H \{aomori, nagano, fukushima\}$.

The concept of an object identifier here is stronger than other approache such as F-logic[KL89]: given two attribute term, $o|C_1$ and $o|C_2$, they should be merged as $o|C_1 \cup C_2$.

### 3.2 QUIK Program

A QUIK program $P$ consists of a data dictionary / directory (DD/D) $D$, mediator relation $M$, subsumption relation $S$, and a set $R$ of rules:

$$P = (D, M, S, R)$$

A QUIK program is defined, corresponding to an (real or virtual) information source as in Figure 1.

$S$ is a lattice of concepts, each of which is a basic object. We can consider a lattice of object terms as a conservative extension of the lattice of basic objects, as in Section 3.1. Later we have to discuss treatment of equivalence relation as a result of generation of a lattice from multiple information sources', differently from $\mathcal{QUIXOTE}$.

A DD/D, $D$, is a set of metadata, which consists of mediator identifiers (MIDs), object identifiers (OIDs), attribute names as follows:

$$D = (ID, \{ID_i, \{O_{ij}, \{L_{ijk}\}\}\}, E)$$

$ID$ is a self-identifier, $ID_i$ is a MID, $O_{ij}$ is an OID included in $ID_i$, $L_{ijk}$ is an attribute name included in $O_{ij}$, and $E$ is a set of equivalence relation. Here we omit physical information such as TCP/IP addresses.

Each rule is in the following form:

$$a \Leftarrow m_1 : a_1, \cdots, m_n : a_n | C$$

$m_i$ is an MID, $a, a_i$ are attribute terms, $C$ a set of constraints. When $n = 0$, a rule is called a fact. By structuring an MID as in a module identifier in $\mathcal{QUIXOTE}$, we can differentiate multiple exchange models on a single information source. As an attribute term, $a$m consists of an object term, $o$, and a set of constraints, $C$, the above rule can be written as follows:

$$o|C \Leftarrow m_1 : o_1, \cdots, m_n : o_n \mid\mid C_1 \cup \cdots \cup C_n \cup C', \text{ or}$$
$$o|C \Leftarrow m_1 : o_1, \cdots, m_n : o_n \mid\mid A \cup C''.$$

The constraint part in the body can be divided into related ones, $C''$, and other objects', $A$. Object terms, $o_1, \cdots, o_n$, are considered as existence checks of corresponding objects, $C$, a set of dotted constraints of $o$, is considered as assertional constraints, $C''$, a set of variable constraints, is considered as a set of constraints to be satisfied, and $A$ is considered as constraints of other objects' extrinsic properties.

The procedural semantics of a specified MID in a body of a rule is the same as the external reference of $\mathcal{QUIXOTE}$, that is, omitted MID is interpreted as its own MID. Information source relation consists of the followings:

| | |
|---|---|
| reference relation: | $I_1 \sqsubseteq_R I_2$ |
| inheritance relation: | $I_1 \sqsubseteq_D I_2$ |

each of which means that $I_1$ refers $I_2$ or $I_1$ inherits $I_2$, respectively.

One of the distinguished features of $\mathcal{QUIXOTE}$ and QUIK is the declarative semantics, which is based on ZFC$^-$/AFA, a hyperset theory proposed by P. Aczel [Aczel88]. The domain is a set of labeled graphs in the sense of ZFC$^-$/AFA for treating circular structure [YY90]. Another distinguish feature is the procedural semantics with abduction, which we explain in Section 5.

## 4 Multiple Subsumption Relations

A mediator defines a integrated view from multiple mediators and reduces redundancy and eliminates their inconsistency. To issue a query to multiple mediators, we must generate a global subsumption relation to maintain their consistency by merging them. When an mediator $M$ refers $M_1, M_2, \cdots, M_n$, let their corresponding subsumption relation be $L, L_1, L_2, \cdots, L_n$. Then the subsumption relation of $M$ is generated by $L \cup L_1 \cup L_2 \cup \cdots \cup L_n$. Generally, a union of lattices $L_1$ and $L_2$ is not necessarily a lattice, but a lattice generation algorithm is well known as in [ALN87]. However, as its automatic application might destroy the relation. When a QUIK program detects a circular relation such as $a \sqsubseteq b \wedge b \sqsubseteq a$ from mediators $m_i$ and $m_j$, we consider the following three possibilities in an interactive environment in QUIK.

1. $a \cong b$
2. $m_i.a \cong m_j.b \wedge m_i.b \cong m_j.a$
3. other relations

Depending on users' decision, the resulting equivalence relations are stored in DD/D.

Let $\overline{L_1 \cup L_2}$ be a lattice generated from lattices $L_1$ and $L_2$. When $P = (D, M, L, R), P_i = (D_i, M_i, L_i, R_i)$ $(1 \leq i \leq n)$, and

$$L' = \overline{L \cup L_1 \cup l_2 \cup \cdots \cup L_n}$$

$L'$ is defined as a global lattice among $P, P_1, P_2, \cdots, P_n$: that is, we can define the global consistent lattice in a mediator. It is exported into all lower mediators and used for constraint solving and property inheritance in each information source.

## 5 Query Processing

### 5.1 Deduction with Abduction

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:

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

where the given query is ?-$G_0|C_0$. 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, because the concept of oid in $\mathcal{QUIXOTE}$/QUIK is stronger than other DOOD languages. Remember that a rule is in the following form:

$$o|C \Leftarrow m_1 : o_1, \cdots, m_n : o_n \mid\mid A \cup C''$$

A query is also transformed in the form of ?-$o_1, \cdots, o_n \mid\mid A_0 \cup C_0$, 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 $G'|C' \Leftarrow B \mid\mid A \cup C$, and $\exists\theta\ G\theta = G'\theta$, where $B$ is a set of object terms and $\theta$ is a substitution, the transformed node is:

$$((G_i \cup B)\theta,\ (A_i\theta \setminus C'\theta) \cup A\theta,\ (C_i \cup C \cup C')\theta).^1$$

The derivation image is illustrated as in Figure 2. 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 $(G, A, C \cup C')$.

Conditional query processing is to insert hypotheses incrementally into a mediator to control its process as nested transactions. A query to a QUIK program, $P = (D, M, S, R)$, can have hypotheses: ?-$q$ with $(D', M', S', R')$, which is equivalent to a query ?-$q$ to a mediator, $(D \cup D', M \cup M', S \cup S', R \cup R')$ The following is an example of a query sequence:

| | |
|---|---|
| ?-open($P$). | % Open a mediator named $P$. |
| ?-begin_trans. | % Begin a transaction (level 1). |
| ?-$q_1$ with $H_1$. | % Same as ?-$q_1$ to $P\cup$ $H_1$. |
| ?-begin_trans. | % Begin a transaction (level 2). |
| ?-$q_2$ with $H_2$. | % Same as ?-$q_2$ to $P\cup$ $H_1\cup$ $H_2$. |
| | % Hypotheses are incrementally inserted. |
| ?-abort_trans. | % Abort a transaction (level 2). |
| | % $H_2$ is rolled back. |
| ?-$q_3$ with $H_3$. | % Same as ?-$q_3$ to $P\cup$ $H_1\cup$ $H_3$. |
| ?-end_trans. | % Commit a transaction (level 1). |
| | % $P$ is updated to $P\cup$ $H_1\cup$ $H_3$. |
| ?-close($P$). | % Close a mediator $P$. |

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

Query processing with hypothesis generation and conditional query is effective for a partial information database[YNTT94]. Differently from convention constraint logic programming languages, a derivation sequence for hypothesis generation is a triple as follows:

$$(G_0, A_0, \emptyset) \Rightarrow (G_1, A_1, C_1) \Rightarrow \cdots \Rightarrow (\emptyset, A_n, C_n)$$

where $G_i$ is a set of object terms to be solved, $A_i$ is a set of constraints to be solved, and $C_i$ is a set of solved constraints. When a set of object terms is satisfied, a set, $A_n$, of unsatisfied constraints is generated as a set of hypotheses and an answer, $(\emptyset, A_n, C_n)$, is returned. It does not take an existence of an object as a hypothesis to reduce a set of answers.

As, in distributed environments, an object might exist in another mediators, relaxation of such restrictions would be effective for many applications. Here, we consider such relaxation in a hierarchical mediator system.

Let $(D, M, S, R)$ be a QUIK program, and $N_i = (G_i, A_i, C_i)$ be an unsatisfied terminal node in a derivation sequence for a query ?-$G$. Let a DD/D, $D$, return a set of MIDs. When $G_i = G'_i \cup \{m_j : o\}, m_j : o \notin G'_i$ and $D$ returns $\{m_1, m_2, \cdots, m_n\}$ for $o$, we can generate the followings from $N_i$:

$$\begin{aligned} N_{i_1} &= (G'_i \cup \{m_1 : o\}, A_i, C_i) \\ N_{i_2} &= (G'_i \cup \{m_2 : o\}, A_i, C_i) \\ &\vdots \\ N_{i_n} &= (G'_i \cup \{m_n : o\}, A_i, C_i) \end{aligned}$$

Further, after satisfying existence of objects, constraints about attributes might be solved by using the DD/D. Variables for information sources can be bound by using the DD/D.

1. $(\emptyset, \emptyset, C_n)$: success
2. $(\emptyset, A_n, C_n)$:
   (a) Search another information source for $A_n$ by using DD/D
   (b) Sccording to the result, repeat 1 or 2.
   (c) If $A_n$ is not satisfied, success with hypotheses
3. $(G_n, A_n, C_n)$:
   (a) Search another information source for $G_n$ by using DD/D
   (b) If $G_n = \emptyset$, then go to 1 or 2
   (c) When $G_n$ is unsatisfied, repeat 3 for $(G_{n-1}, A_{n-1}, C_{n-1})$
   (d) If $n = 0$, then failure

$$\begin{tikzcd}
 & \vdots & \cdots \arrow[r] & (G_i,A_i,C_i) \arrow[dr] & & & \\
(G_0,A_0,C_0) \arrow[urr] \arrow[drr] & & & & (G_k,A_k,C_k) \arrow[r] & \cdots \arrow[r] & (\emptyset,A_n,C_n) \\
 & \vdots & \cdots \arrow[r] & (G_j,A_j,C_j) \arrow[ur] & & &
\end{tikzcd}$$

Figure 2: Derivation Network

For example, consider the following three mediators:

$$\begin{array}{ll}
m_1: & q/[l=a] \\
m_2: & r \\
m_3: & p/[l=X] \Leftarrow m_1 : q[l=X], m_2 : r/[l=X]
\end{array}$$

and a query ?-$m_3$ : $p$. If $m_3$ does not have any information about objects in other mediators, $m_3$ returns an answer, "If $m_2$ : $r/[l=a]$, then yes." If DD/D in $m_3$ knows which mediators have $l$-property of $o$, an alternative query, ?-$m_X$ : $r/[l=a]$ to a mediator $m_X$ as in Figure 3.

As for the existence on an object, consider the following:

$$\begin{array}{lll}
m: & (m, \{(m_1, \{q, r\}), (m_2, \{r, s\}), & \text{DD/D} \\
 & \quad (m, \{p, \cdots\})\}, \{\}) & \\
 & \{m \sqsubseteq_R m_1, m \sqsubseteq_R m_2\} & \text{Mediator rel.} \\
 & p \Leftarrow m_1 : q, m_2 : r. & \\
m_1: & q. & \\
 & r. & \\
m_2: & r \Leftarrow s. &
\end{array}$$

Although a query ?-$p$ for a mediator $m$ fails, we can get candidate answers by an alternative subgoals of $m_1$ : $r$, because the DD/D knows that $r$ exists in $m_1$.

### 5.2 Extensions of Query Processing

First, we have to consider dispatching mechanism among mediators. In Section 5.1, sub-queries are sent according to mediator identifiers (MIDs) specified in the program. It is not flexible to specify MIDs in a program explicitly, and it is difficult to restrict search space, if variables are used as MIDs. So we allow to use function names instead of MIDs as in [AY95], which are controlled by DD/D. Function names are specified in a newly introduced self-model in a QUIK program.

During the initialization of mediators, its corresponding mediator constructs a map, called a mediator directory, of the mediator name and its physical address (IP address, process identifier, and so on). Then the mediator gathers information on objects and attributes, and information on functions provided as function names in the self-model. Then two kinds of maps are constructed: a map of objects, attribute names, and mediator names called an object directory, and a map of function names and mediator names called a function directory. Such maps are used in dispatching sub-queries between mediators.

Since a function name does not necessarily correspond to a mediator uniquely, a message is possibly sent to multiple mediators if a function name is used to designate destination mediators. This mechanism is useful for the following reasons:

- It is unnecessary to specify a destination mediator name explicitly.
- It is possible to send a message simultaneously to candidate mediators.

A mediator decides to send sub-queries sequentially or simultaneously to candidate mediators listed by the maps, and processes answers sequentially or grouped as a set.

Secondly, we plan to introduce multiagent-based cooperation mechanisms such as contract net[Smith80], because some mediators might not have all information about objects and could not export their meta information because of their high autonomy. In such a case, other mediators can user only function names.

For example, the contract net protocol was proposed as one of the protocols for a manager and contractors to dispatch a job by the manager to a contractor. This protocol consists of *Task announcement*, *Bidding*, *Announcement of award*, and so on. Basically, a mediator announce a task to candidate mediators, get and evaluate their bids, and send a query to selected mediators. This protocol is effective in the distributed environment where there are many autonomous information sources that do not export their contents to their corresponding QUIK programs.

Generally, we consider to introduce other cooperation protocols and strategies for more flexible query processings.

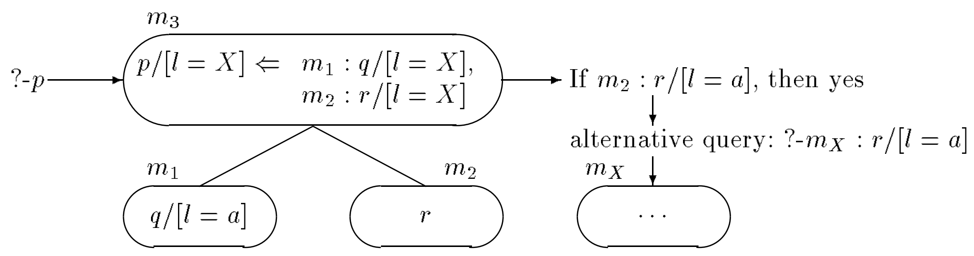
$m_3$
$?\text{-}p$
$p/[l = X] \Leftarrow m_1 : q/[l = X], m_2 : r/[l = X]$
If $m_2 : r/[l = a]$, then yes
alternative query: $?\text{-}m_X : r/[l = a]$
$m_1$
$m_2$
$m_X$
$q/[l = a]$
$r$
$\cdots$

Figure 3: Abduction in Mediators

## 6 Conclusions

We are now implementing a QUIK system in Java and designing various application such as a literature database and workflow management. As a literature database, we take the Deidre Legend, one of the most famous Irish literature. In this application, we show that QUIK mediators work effectively not only for distributed information sources, but also for multiple layers from one source. As workflow management, we show that QUIK mediators could serve as an environment of computer supported cooperative works, because a QUIK mediator encapsulates also a human.

In this paper, we omit transformation between a QUIK program and an information source. However we are designing some algorhims for the above applications, but not for general sources.

As mentioned in Section 1, information sources in distributed environments are autonomous and sometimes unstable. We have to consider the following dynamic aspects:

- When does a DD/D get new information about objects and their attributes under its information source relation?
- When does a QUIK mediator inherit its related rules?
- When is the global subsumption relation generated?

For such purposes, we are considering refreshing on demand, automatic reflection, and instantiation of mediators.

Further, we are discussing an extended strategy for searching unsatisfied objects to upper mediators and its interactive environment.

In this paper, we introduce a new language QUIK and describe its query processing facilities as an extension of an mediator architecture as follows:

- We propose a new knowledge representation language QUIK as an extension of a DOOD language *Quixote*. We extend an exchange model by introducing rules and subsumption relation.
- We can define multiple exchange models of a single information source by structuring an MID.
- With capabilities of not only mediator specification but also hypothesis generation, a mediator can search alternative information sources automatically.
- By generating the global subsumption relation and export it to related mediators, and by controlling syntactical differences among OIDs, we can reduce inconsistencies among mediators.

## References

[AY95] A. Aiba, K. Yokota, and H. Tsuda, "Heterogeneous Distributed Cooperative Problem Solving System Helios and Its Cooperation Mechanisms", *Int. J. of Cooperative Information Systems*, pp.369-385, vol.4, no.4, Dec., 1995.

[Aczel88] P. Aczel, *Non-Well-Founded Set Theory*, CSLI Lecture Notes No. 14, 1988.

[ALN87] H. Aït-Kaci, R. Boyer, P. Lincoln, and R. Nasr, "The Efficient Implementation of Object Inheritance", *MCC Tech. Report AI-102-87*, 1987.

[KL89] 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.

[PAG96] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina, "Object Fusion in Mediator Systems", In Proc. Int. Conf. Very Large Data Bases, India, Aug., 1996

[Smith80] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transaction on Computers*, Vol. C-29, No.12, pp.1104 - 1113, December 1980.

[YY90] H. Yasukawa and K. Yokota, "Labeled Graphs as Semantics of Objects", Proc. Joint Workshop of SIGDBS and SIGAI of IPSJ, Nov., 1990.

[YY92] 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 Fifth Generation Computer Systems (FGCS'92)*, pp. 89-112, Tokyo, June, 1992.

[YNTT94] K. Yokota, T. Nishioka, H. Tsuda, and S. Tojo, "Query Processing for Partial Information Databases in *QUIXOTE*", *6th IEEE International Conference on Tools with Artificial Intelligence (TAI'94)*, pp.359-365, New Orleans, Nov., 1994.

[Yokota94] K. Yokota, "From Databases to Knowledge-Bases — Kappa, *QUIXOTE*, Helios", *Proc. Int. Symp. on Fifth Generation Computer Systems (FGCS'94)*, pp.34-50, Tokyo, Dec., 1994.