ICOT Technical Report: TR-223

TR-223

Partial Evaluation of Knowledge Base as Specification for Queries

by

C. Sakama and H. Itoh

November, 1986

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

Partial Evaluation of Knowledge Base as Specification for Queries

Chiaki Sakama and Hidenori Itoh

Institute for New Generation Computer Technology
Mita Kokusai Building 21F, 1-4-28 Mita, Minato-ku, Tokyo 108 Japan.

December 15, 1986

Abstract

This paper presents a partial evaluation method as a specification technique for a knowledge base for given queries. It is a transformation of a knowledge base into a virtual relation which preserves the equivalence of the queries. It is effective when some iterative query processing occurs on a large scale of knowledge base.

keywords : Partial evaluation, Knowledge base, Least generalized query, Virtual knowledge base.

1 Introduction

We are now developing some knowledge base systems which can deal with a large amount of knowledge effectively. And in a large scale of knowledge base, it is important how to process some queries on it effectively. Searching strategies and knowledge structuring methods have been discussed for this purpose.

Certainly, considering a database as a virtual relation enables us to utilize the data effectively. In the case of a knowledge base, it is considered as a virtual relation derived from queries through deduction. But when there are a lot of queries to process, it is ineffective to derive answers for every query iteratively because there may be similar deduction steps in their retrieval. So we consider specifying a knowledge base for the given queries in advance by partially evaluating the knowledge base under the environment of the queries and optimizing their retrieval.

Partial evaluation of programs is effective in iterative computation [Futamura 83], and its applications in logic programming [Takeuchi 85] and deductive databases [Miyazaki 86] have been discussed. In this paper we assume a knowledge base represented by a logic programming language such as Prolog and describe its partial evaluation as transformation of a knowledge base into a virtual relation for given queries which preserves their equivalence.

2 Partial evaluation of knowledge base

Suppose there is a knowledge base composed of Horn clauses and a set of queries. In this section we discuss the partial evaluation method of a knowledge base for effective query processing.

2.1 Least generalized query

First we note the concept of least (common) generalization of terms which is the dual notation of the greatest (common) instance[Reynolds 70].

Definition (Least generalization)

1. Given terms P and Q, Q is more general than P iff there exists a substitution θ such that $P = Q\theta$, written $P \sqsubseteq Q$.
2. Let S be a set of terms, then L is a *least generalization* of S iff

i) $\forall T \in S, T \sqsubseteq L$

ii) $\forall T \in S, \forall L_i$ *s.t.* $T \sqsubseteq L_i, L \sqsubseteq L_i$ □

That is, L is the least upper bound of the lattice of S.

Example The least generalizations of

$\{f(a,a)), f(a,g(b))\}$ and $\{f(a,g(X)), f(h(X),g(h(X))), f(g(X),g(Y))\}$

are $f(a,X)$ and $f(X,g(Y))$, respectively. □

Using this notation, we define the *least generalized query* (LGQ for short) as a least generalization of the given set of queries. [1]

[1] We assume a query composed of a single literal becase a query composed of several literals can be transformed into a single one by asserting a transformation rule. For example, a query ?- f(X,Y),g(Y,Z),h(Z). is transformed into a query ?- p(X,Y,Z). and a rule p(X,Y,Z):- f(X,Y),g(Y,Z), h(Z). etc.

The algorithm for getting the LGQ from the given query set is as follows.

i) Classify the queries into sets of queries which have the same predicate symbols and the same number of arguments.

ii) Apply the least generalization algorithm to all the classified sets of queries.

The least generalization algorithm[Reynolds 70] is as follows;

Let A and B be any two predicates which have the same predicate symbols and the same number of arguments, and $Z_1, Z_2, \ldots$ be a sequence of variables which don't occur in A or B.

Step.1 Set the variables A' to A, B' to B, ς and η to empty substitution, and i to zero.

Step.2 If $A' = B'$, exit with $A \sqcup B = A' = B'$. [2]

Step.3 Let k be the index of the first symbol position at which A' and B' differ, and let S and T be the terms which occur, beginning in the kth position, in A' and B' respectively.

Step.4 If, for some j such that $1 \leq j \leq i$, $Z_j\varsigma = S$ and $Z_j\eta = T$, then alter A' by replacing the occurance of S beginning in the kth position by Z_j, alter B' by replacing the occurence of T beginning in the k th position by Z_j, and go to 2.

Step.5 Otherwise, increase i by one, alter A' by replacing the occurance of S beginning in the kth position by Z_i, alter B' by replacing the occurance of T beginning in the kth position by Z_i, replace ς by $\varsigma \cup S/Z_i$, replace η by $\eta \cup T/Z_i$, and go to 2.

This algorithm always terminates at step2, and for the finite set of predicates $\{P_1, P_2, \ldots, P_n\}$, which have the same predicate symbol and the same number of the arguments, we can get a least generalization of this set by using this algorithm for each predicate iteratively.

Example The LGQ of the query set

$\{\neg f(X, h(a)), \neg f(b, h(X)), \neg g(h(X), Y, Y), \neg g(a, h(X), h(X)), \neg h(a), \neg h(h(b))\}$

is

$\{\neg f(X, h(Y)), \neg g(X, Y, Y), \neg h(X)\}$ $\square$

Next we consider specification of a knowledge base by such an LGQ.

[2] $\sqcup$ denotes the least upper bound.

2.2 Virtual knowledge base

We want to reduce the search space for the given queries by partially evaluating a knowledge base in advance using them. For this purpose, we consider deriving an *LGQ* of the given queries and evaluating a knowledge base using it to create a virtual relation for the queries.

We call the virtual relation a *virtual knowledge base* which is a specificated knowledge base for the given queries.

The step of creating a virtual knowledge base is as follows:

Step.1 Analyze given queries and derive an *LGQ*.

Step.2 Retrieve the *LGQ*'s answers from the knowledge base.

A virtual knowledge base is the instantiated *LGQ*s by its answers and retrieval for the given queries are done in the resultant virtual table.

Example Suppose the knowledge base is given.

```
p(X,Y,Z):- q(X,Y),r(Y,Z).
q(X,Y):- w(X,Z),q(Z,Y).
q(X,Y):- w(X,Y).
r(X,Y):- s(X,f(Y)),t(Y).
u(X,Y):- m(X,Y),t(Y).
m(X,Y):- n(c,X,Y).
l(X):- t(X),v(X).
s(b,f(g(c))).  s(d,f(a)).  s(f(a),c).  s(c,f(f(b))).  s(g(a),f(f(g(b)))).  s(f(d),f(c)).
t(a).  t(b).  t(f(b)).  t(f(g(b).  t(g(c)).
v(a).  v(b).  v(f(c)).
w(f(a),b).  w(f(a),c).  w(f(f(c)),d).  w(g(b),c).
n(c,a,b).  n(c,a,f(g(b))).  n(c,a,g(c)).  n(c,b,g(d)).  n(b,c,f(d)).
```

Consider a set of queries

$$\mathbf{q} = \{\neg p(f(X),Y,Z), \neg p(f(X),Y,X), \neg p(f(f(X)),g(X),Y),$$
$$\neg u(a,b), \neg u(a,c), \neg r(a,f(X)), \neg r(X,f(a))\}$$

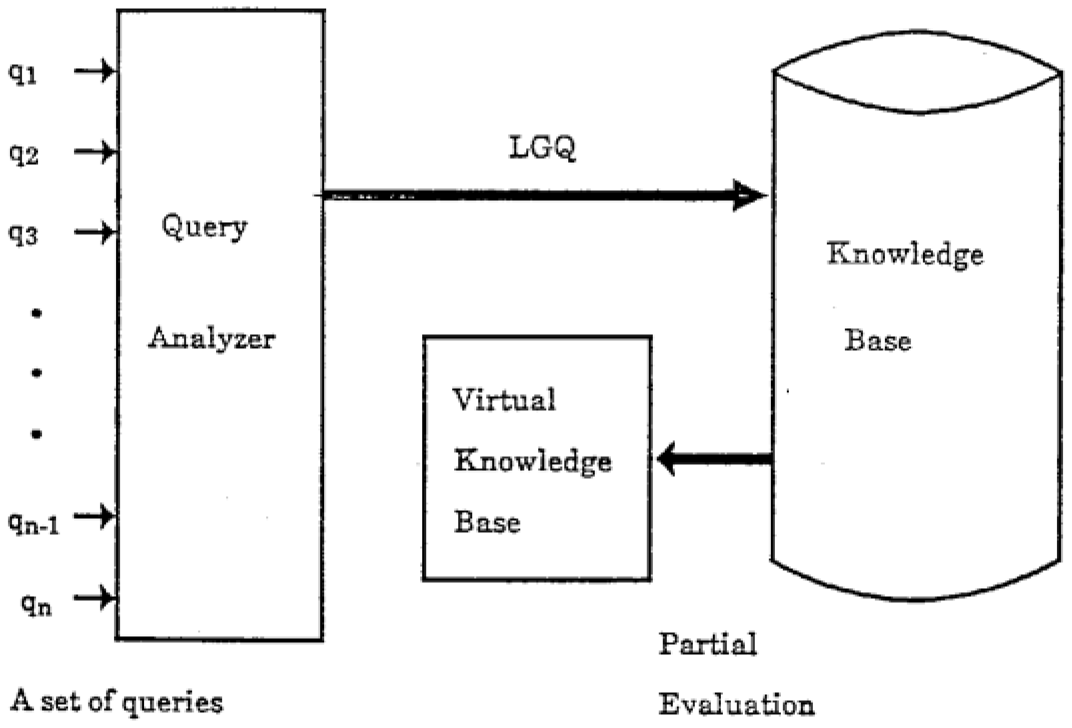

Figure 1. Model of creating virtual knowledge base

then the *LGQ* of **q** is

$$LGQ = \{\neg p(f(X), Y, Z), \neg u(a, X), \neg r(X, f(Y))\}.$$

and the virtual knowledge base is as followes;

p(f(a),b,g(c)). p(f(a),c,f(b)). p(f(f(c)),d;a).
u(a,b). u(a,f(g(b))). u(a,g(c)).
r(c,f(b)). r(g(a),f(g(b))). □

In other words, by creating the virtual knowledge base in advance, we can omit iterative deduction for the given queries. Our model of creating a virtual knowledge base is shown in Figure 1.

Query processing in this model is presented as follows.

Partial evaluation of a knowledge base kb by LGQ is denoted

$$kb_{LGQ} = \Re(kb, LGQ) \tag{1}$$

where $\Re$ is a function which accepts a knowledge base and queries, returns instantiated queries with their answers, and kb_{LGQ} is the evaluated kb by LGQ, that is, a virtual relation of LGQ.

LGQ, derived by query analysis from a set of queries $\mathbf{q}$ is

$$LGQ = \sqcup \mathbf{q} \tag{2}$$

Applying (2) to (1),

$$kb_{LGQ} = \Re(kb, \sqcup \mathbf{q}) \tag{3}$$

Using (3), the retrieval of $\mathbf{q}$ in kb_{LGQ} is denoted as follows :

$$\Re(kb_{LGQ}, \mathbf{q}) = \Re(\Re(kb, \sqcup \mathbf{q}), \mathbf{q}) \tag{4}$$

(4) denotes a partial evaluation of knowledge base and the query processing in our model. It is obvious that this transformation of kb into kb_{LGQ} preserves equivalence for the set of queries $\mathbf{q}$, because kb_{LGQ} is the virtual relation of the set of answers of the LGQ and includes the answers of the set of queries.

3 Discussion

This section gives some results of the experimental implementation of query analysis and discusses the conditions for the effectiveness of our model by estimating the performance of our model.

3.1 Experimental results

Our model for query processing consists of these three processes :

i) Query analysis

ii) Creating a virtual knowledge base

iii) Query processing in the virtual knowledge base

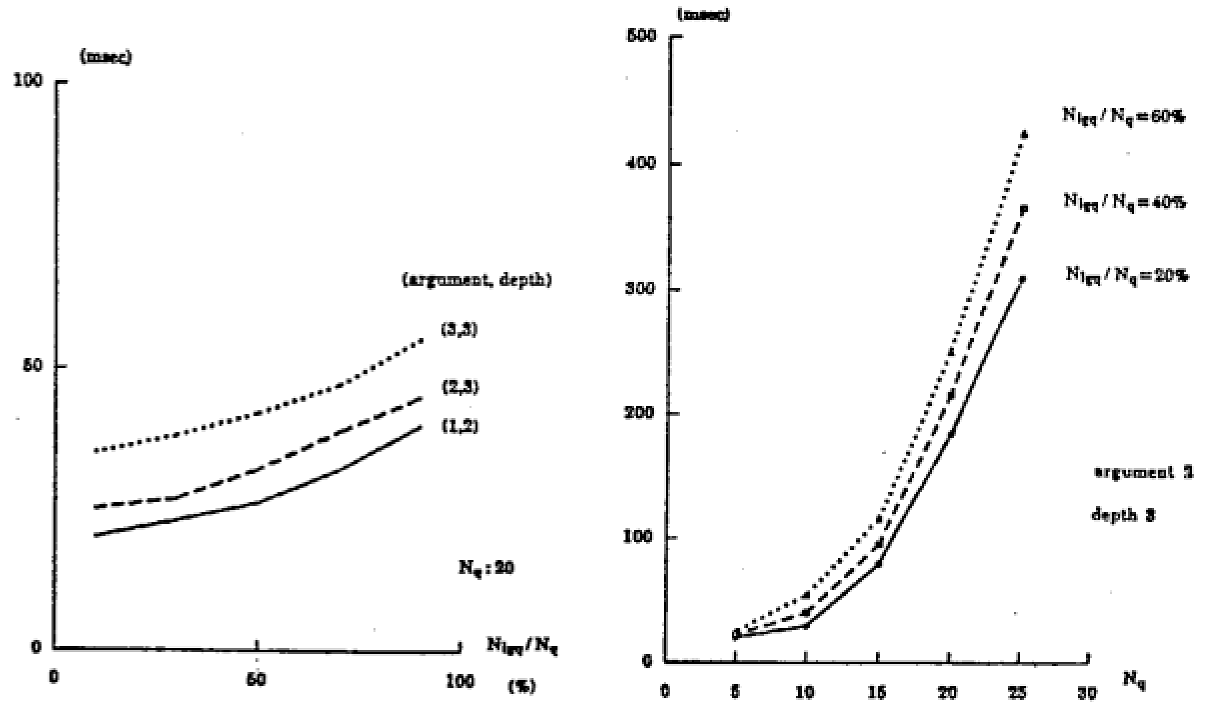

Figure 2. Query analysis results

First some results of query analysis are shown in Figure 2. It depends on the structure of queries(i.e the number of arguments and the depth of terms), the number of queries to analyze and the rate of N_{LGQ}/N_q, where N_{LGQ} and N_q denote the number of LGQ and queries, respectively.

3.2 Estimated performance

Next we estimate the execution time of query processing in our model.

When a Prolog compiler is $C\ LIPS$, then the execution time of query procesing is given by $N_a \cdot R/C$ where N_a and R denote the number of answers for a query and its average reduction steps for one answer, respectively. When the number of queries to retrieve is N_q then the execution time of these queries is given by $N_q \cdot N_a \cdot R/C$, where $N_a \cdot R/C$ is the average execution time for the queries.

Assume the number of LGQ is N_{lgq} and the average execution time of them is $N_{lga} \cdot R_{lgq}/C$, the time for creating a virtual knowledge base is given by $N_{lgq} \cdot N_{lga} \cdot R_{lgq}/C$. Then the execution time of query processing in the virtual knowledge base is $N_q \cdot N_a/C$, $R = 1$ above is since the virtual knowledge base is a relation for the queries.

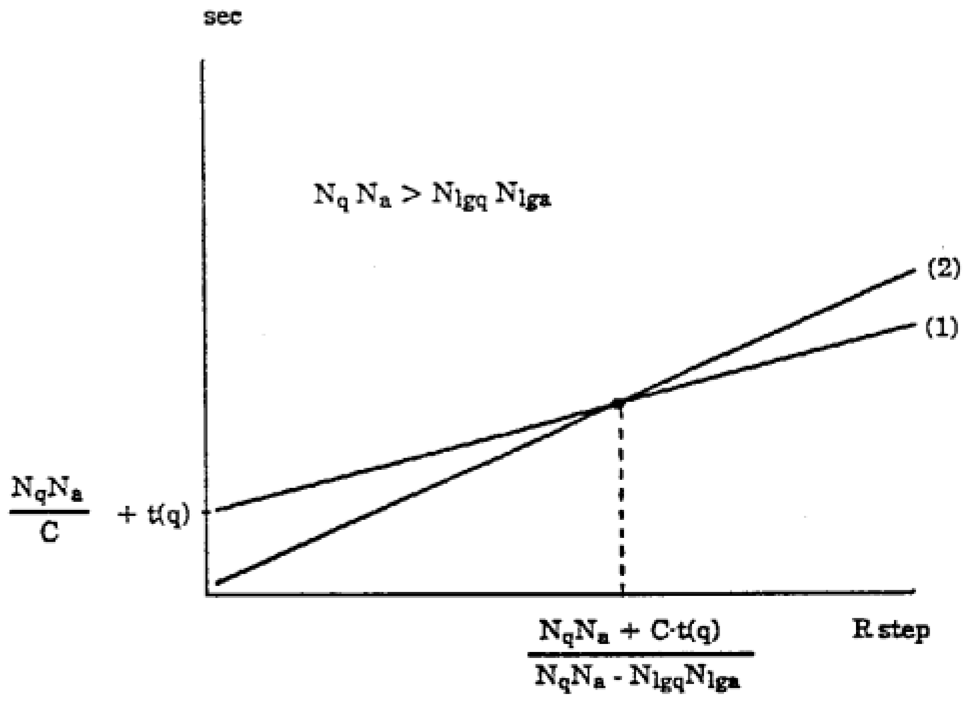

Figure 3. Comparison of execution time

Since the total execution time of query processing is

$$N_{lgq} \cdot N_{lga} \cdot R_{lgq}/C + N_q \cdot N_a/C + t(\mathbf{q}) \tag{1}$$

$t(\mathbf{q})$ above is a time for query analysis which depends on the property of the set of queries.

On the other hand, execution time of query processing in a knowledge base is given by the following:

$$N_q \cdot N_a \cdot R_q/C \tag{2}$$

where R_q denotes the average reduction steps for queries.

Now we compare the above two execution time of query processing.

Figure 3 shows the comparison of the above two execution time. Here we assume R_q is nearly equal to R_{lgq} under the condition of compiling with indexing[Bowen 83], and both are presented as a parameter R. From Figure 3, when $N_{lgq} \cdot N_{lga}$ is smaller than $N_q \cdot N_a$, it takes less execution time in our model beyond $N_q \cdot N_a + C \cdot t(\mathbf{q})/N_q \cdot N_a - N_{lgq} \cdot N_{lga}$ average reduction steps.

For example, assume $C = 43000 LIPS$ (DEC 2060)[Kaneda 84], $N_q = 20, N_{lgq} = 8$ ($N_{lgq}/N_q =$

40%) and $N_a = 10, N_{lga} = 15$, by the Figure 2 we get $t(\mathbf{q}) = 220msec$, and with these values the above average reduction step is calculated about 121.

Finally, we summerize the conditions when our model is effective as follows.

(*a*) There is a large scale of knowledge base and takes many reduction steps to process queries in them and further the load for query processing in them is much heavier than that of query analysis. (Notice the query analysis $t(\mathbf{q})$, depends on the queries and not on the knowledge base.)

(*b*) There is considerable similar iterative query processing. (i.e $N_{lgq}/N_q \ll 1$.)

(*c*) The answers approximated by *LGQ* contains few excessive answers for the queries or the queries have some answers in common. (In the latter case, *LGQ* omits the redundant answers.)

4 Concluding remarks

In this paper we discussed partial evaluation of a knowledge base as a transformation of the knowledge base into a virtual knowledge base which preserves equivalence of the given queries. This technique is considered effective for example when a large amount of knowledge is stored in a secondary memory and its access by some iterative query processing happens frequently. In this case creating a virtual knowledge base is considered as a preprocess of compiling the knowledge by *LGQ* in advance and storing it in the primary memory as a virtual relation for the queries.

In order to utilize a large amount of knowledge effectively, it is available to modify knowledge base under the environment of usage, and we consider this research as one of the techniques for that.

Acknowledgments

The authors would like to thank their colleagues in the KBM project for many useful discussions.

References

[Futamura 83] Futamura,Y.: "Partial Computation of Programs", Jounal of IECE of Japan, vol.66, No.2, 1983.

[Takeuchi 85] Takeuchi,A. and Furukawa,K.: "Partial Evaluation of Prolog Programs and its Application to Meta Programming", Proc. of the Logic Programming Conference'85, ICOT, 1985.

[Miyazaki 86] Miyazaki,N., Yokota,H. and Itoh,H.: "Compiling Horn Clause Queries in Deductive Databases: A Horn Clause Transformation Approach", ICOT Technical Report, TR-183, 1986.

[Reynolds 70] Reynolds,J.C.: "Transformational Systems and the Algebraic Stracture of Atomic Formulas", Machine Intelligence, vol.5, pp135-151, 1970.

[Bowen 83] Bowen,D.L.: "DECsystem-10 PROLOG USER'S MANUAL", University of Edinburgh, Dept. of Artificial Intelligence, 1983.

[Kaneda 84] Kaneda,Y.: "Prolog Machine", Jounal of IPS of Japan, vol.25, No.12, 1984.