ICOT Technical Report: TR-149

TR-149

Formal Semantics of a Relational Knowledge Base

by
Masaki Murakami, Haruo Yokota
and
Hidenori Itoh

December, 1985

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

Formal Semantics of a Relational Knowledge Base

Masaki Murakami, Haruo Yokota, Hidenori Itoh

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

December 1985

ABSTRACT

A mathematical foundations for formal semantics of term relations [Yokota et al. 85] is presented. A term relation is a basic data structure of a relational knowledge base. It is an enhanced version of relational model in a database theory. It may include syntactically complex structures such as terms or literals containing variables as items of relations. The items are retrieved with operations called *retrieval-by-unification*. We introduce as a semantic domain of n-ary-term relations $n\_T\_RELATIONS$ and define a partial order on them. We characterize *retrieval-by-unification* operations as operations on $n\_T\_RELATIONS$ with monotone functions and greatest lower bounds.

1. Introduction

The Fifth Generation Computer Systems (FGCS) project in Japan aims to develop inference and knowledge base mechanisms to implement a knowledge information processing system. In the intermediate four-year (1985-88) stage of the project, we plan to develop prototypes of knowledge base machines. In the first three-year (1982-84) stage of the project, we developed database machine *Delta* [Kakuta et al. 84] to investigate techniques for fast retrieval from a large amounts of data. But each item in relational databases only represents an element of a relation defined on a finite domain. In other words, it can handle only the fact clauses of Prolog. We need to handle more complex structures including variables such as rules and S-expressions of LISP in our knowledge base system besides fact clauses. In [Yokota et al. 85], we introduce a *relational knowledge base* model. Relational knowledge bases are sets of *term relations*. Each item in a term relation is a term, containing a number of variables. The system presented here handles complex structures of knowledge such as Horn clauses or S-expressions on them. A

number of new operations called *retrieval-by-unification (RBU)* operations to retrieve them are introduced.

A very large knowledge base system are shared among a number of users. The information stored in such knowledge bases is accessed by many users. In order to share software libraries or databases, their specifications must be precise and easy to understand. This is the case for knowledge bases. Therefore, knowledge representation languages must have unambiguous semantics. In this paper, we present mathematical bases for formal semantics of term relations.

Section 2 is an outline of a relational knowledge base with data structures of term relations and RBU operations. We introduce $n\_T\_RELATIONS$ as a semantic domain of n-ary term relations and define a partial order on it in Section 3. It is shown that the domain is a lattice with a partial order. In Section 4, RBU operations are characterized as operations in the semantic domain using monotone functions and greatest lower bounds.

2. Data Structures of Relational Knowledge Base

The basic structures of relational knowledge bases are term relations, i.e., finite sets of tuples of terms.

Def.1 Let *Var* be a set of variables, *Fun* be a set of function symbols. *Fun* is a finite set, *Var* is an enumerably infinite set. Each element of *Fun* has a specific arity.

Def.2 Let *Terms* be a set of terms defined as follows:

(i) if $t \in Var$ or $t \in Fun$ and t is a 0-ary symbol, then $t \in Terms$.

(ii) if $t_1, \ldots, t_n \in Terms (n \geq 1)$, $f \in Fun$ is n-ary function symbol, then

$$f(t_1, \ldots, t_n) \in Terms.$$

Def.3 A mapping $\sigma : Var \rightarrow Terms$ is called a *substitution* if it satisfies the following condition:

$$\text{If } \Sigma = \{< x, \sigma x > | \sigma x \neq x, x \in Var\}, \text{ then } \Sigma \text{ is a finite set.}$$

We expand the domain of substitutions from *Var* to *Terms*.

Def.4 For each $t \in Terms$, σt is recursively defined as follows.

$$\sigma t = \begin{cases} \sigma x & : t = x \in Var \\ c & : t = c \in Fun \text{ and is a 0-ary symbol} \\ f(\sigma t_1, \ldots, \sigma t_n) & : t = f(t_1, \ldots, t_n), f \in Fun, t_i \in Terms \end{cases}$$

We treat an n-ary tuple of terms as a kind of term. The principal functor of the term is a special function symbol not in Fun.

Def.5 Let $t_1, \ldots, t_n \in Terms$, f is a special n-ary function symbol not in Fun.

$$f(t_1, \ldots, t_n)$$

is called an n-ary tuple. This is usually written simply $(t_1, \ldots, t_n)$.

We use the notation n_Tuples for the set of all n-ary tuples.

The definition of a substitution is extended to a mapping $n\_Tuples \rightarrow n\_Tuples$. We can define a unification between tuples in a similar way for the case of terms.

Def.6 For $t = (t_1, \ldots, t_n), t' = (t_1', \ldots, t_n') \in n\_Tuples$, t and t' are *unifiable* iff there exist a substitution σ such that,

$$(\sigma t_1, \ldots, \sigma t_n) = (\sigma t_1', \ldots, \sigma t_n').$$

The most general unifier of two tuples t_1 and t_2 is written $mgu(t_1, t_2)$.

Def.7 Let T be a finite subset of n_Tuples, T is called an n-ary *term relation*. $n\_T\_Relations$ is the sets of all n-ary term relations.

Def.8 A mapping p is called an *extended permutation* when p is a (partial) one-to-one mapping on integers, e.g. from $\{1, \ldots, n\}$ to $\{1, \ldots, m\}$. $e\_perm_{n,m}$ is the set of all extended permutations from $\{1, \ldots, n\}$ to $\{1, \ldots, m\}$.

Def.9 For a given $p \in e\_perm_{n,m}$, the mapping r_p:$n\_Tuples \rightarrow m\_Tuples$ defined as follows, is called a *reconfiguration* .

$$r_p(t_1, \ldots, t_i, \ldots, t_n) = (t_1', \ldots, t_j', \ldots, t_m')$$

where $t_j' = t_i$ if $p(i) = j$, otherwise $t_j' = x$, where x is a new variable that dose not occur elsewhere in $(t_1', \ldots, t_j', \ldots, t_m')$.

Now we define *retrieval-by-unification*(RBU) operations on a set of term relations.

Def.10 (1) For a given $p \in e\_perm_{n,m}$, the *projection* operation $project_p$:

$n\_T\_Relations \to m\_T\_Relations$ is defined as follows.

$$project_p(T) = \{t | t = r_p(t'), t' \in T\}$$

(2) The *unification_restriction* operation $u\_restriction_i$:

$1\_T\_Relations \times n\_T\_Relations \to n\_T\_Relations$ is defined as follows:

$$u\_restriction_i(T_1, T_2) = \{t | t_1 \in T_1, t_2 \in T_2, t = mgu(r_p(t_1), t_2)\}$$

where $p \in perm_{1,n}$, $p(1) = i$ and $n \geq i \geq 1$.

(3) The *(natural) unification_join* operation $u\_join_{i,j}$:

$n\_T\_Relations \times m\_T\_Relations \to n + m - 1\_T\_Relations$ is defined as follows:

$$u\_join_{i,j}(T_1, T_2) = \{t | t_1 \in T_1, t_2 \in T_2, t = mgu(r_p(t_1), r_q(t_2))\}$$

where $p \in perm_{n,n+m-1}, p(k) = k (n \geq k \geq 1)$ and $q \in perm_{m,n+m-1}$,

$$q(l) = \begin{cases} n + l & \text{if } l < j, \\ i & \text{if } l = i, \\ n + l - 1 & \text{if } l > j. \end{cases}$$

Example.1 Let $f, g \in Fun$ be 1-ary symbols, and $a, b \in Fun$ be 0-ary symbols,

$$x_k, y_k, z_k, u_k, v_k \in Var \text{ for } k = 1, 2.$$

For

$$T_1 = \{(f(x_1), a), (f(g(y_1)), z_1), (g(f(u_1)), a), (g(v_1), v_1)\},$$

$$T_2 = \{(g(f(b)))\},$$

$$T_3 = \{(a, f(x_2)), (f(y_2), y_2)\},$$

$$project_p(T_1) = \{(f(x_1)), (f(g(y_1))), (g(f(u_1))), (g(v_1))\}$$

$$u\_restriction_1(T_2, T_1) = \{(g(f(b)), a), (g(f(b)), f(b))\}$$

$$\begin{aligned} u\_join_{2,1}(T_1, T_3) = \{&(f(x_1), a, f(x_2)), (f(g(y_1)), a, f(x_2)), \\ &(g(f(u_1)), a, f(x_2)), (g(a), a, f(x_2)), \\ &(f(g(y_1)), f(y_2), y_2), (g(f(y_2)), f(y_2), y_2)\} \end{aligned}$$

where $p \in e\_perm_{2,1}$, and $p(2) = 1$.

In [Yokota et al.85], as an application of RBU operations, it was shown that resolutions in logic programming are performed by storing a set of Horn clauses in a term relation. A (definite) Horn clause in DEC-10 Prolog syntax

$$P : -Q_1, Q_2, \ldots, Q_n$$

is stored in a term relation as following 2-ary tuple of following form:

$$(cons(P, X), cons(Q_1, cons(Q_2, cons(\ldots, cons(Q_n, X))\ldots)))$$

where *cons* is a new function symbol for making LISTs and X is a new variable. In the syntax of DEC-10 Prolog, it can be written as follows.

$$([P|X], [Q_1, Q_2, \ldots, Q_n|X])$$

Let D be a term relation that stores definite clauses thus:

$$\begin{aligned}&\{([P_1|X_1], [Q_{1,1}, Q_{1,2}, \ldots, Q_{1,n_1}|X_1]),\\ &\ ([P_2|X_2], [Q_{2,1}, Q_{2,2}, \ldots, Q_{2,n_2}|X_2]), \ldots,\\ &\ ([P_m|X_m], [Q_{m,1}, Q_{m,2}, \ldots, Q_{m,nm}|X_m])\}\end{aligned}$$

and G be a term relation representing a goal clause.

$$G = \{([G_1, G_2, \ldots, G_l])\}$$

Let the result of unification_join of D and G be T_1. T_1 is as follows.

$$T_1 = u\_join_{1,1}(D, G) =$$

$$\begin{aligned}&\{([mgu(P_1, G_1), \sigma_1 G_2, \ldots, \sigma_1 G_l], [\sigma_1 Q_{1,1}, \sigma_1 Q_{1,2}, \ldots, \sigma_1 Q_{1,n_1}, \sigma_1 G_2, \ldots, \sigma_1 G_l]),\\ &\ ([mgu(P_2, G_1), \sigma_2 G_2, \ldots, \sigma_2 G_l], [\sigma_2 Q_{2,1}, \sigma_2 Q_{2,2}, \ldots, \sigma_2 Q_{2,n_2}, \sigma_2 G_2, \ldots, \sigma_2 G_l]), \ldots,\\ &\ ([mgu(P_m, G_1), \sigma_m G_2, \ldots, \sigma_m G_l], [\sigma_m Q_{m,1}, \sigma_m Q_{m,2}, \ldots, \sigma_m Q_{m,nm}, \sigma_m G_2, \ldots, \sigma_m G_l])\}\end{aligned}$$

where σ_i is a substitution such that

$$\sigma_i P_i = \sigma_i G_1 = mgu(P_i, G_1).$$

The second attribute of each element of T_1 corresponds to the resolvent derived by the first step of input resolution. Next we invoke the unification_join operation :

$$T_2 = u\_join_{1,1}(D, project_p(T_1))$$

where $p \in e\_perm_{2,1}, p(2) = 1$. Repeating this, we continue to derive resolvents. The empty clause is arrived at when the second attribute of some tuple is an empty list. If we record the sequence of substitutions through this process, we can derive the result directly by applying the composition of the substitutions in the correct sequence to the variables occurring in the goal clause.

3. Semantics

[Yoshida et al.85] showed that the semantic domain of terms is a complete lattice. This result can be extended to *n_Tuples*. First define the semantic domain of *n_Tuples*, a complete lattice with a partial order.

Def.11 A substitution σ is called a *renaming*, when it is a one-to-one mapping on *Var*. The set of all renaming is denoted by E. The binary relation $\sim$ on *n_Tuples* is defined as follows.

Def.12 For any $t_1, t_2 \in n\_Tuples$,

$$t_1 \sim t_2 \text{ iff } \exists \lambda \in E, t_1 = \lambda t_2.$$

It is easy to show that $\sim$ is an equivalence relation on *n_Tuples*. A semantic domain *n_TUPLES°* of *n_Tuples* is a set of equivalence classes of $\sim$.

Def. 13

$$n\_TUPLES \equiv \{C_r(a) | a \in n\_Tuples\}$$

$$n\_TUPLES^\circ \equiv n\_TUPLES \cup \{\bot_n\}$$

where

$$C_r(a) \equiv \{t | t \in n\_Tuple, t \sim a\}.$$

We define the binary relation $\sqsubseteq$ on *n_TUPLES°* as follows.

Def.14 For any $r_1, r_2 \in n\_TUPLES^\circ$,

$$r_1 \sqsubseteq_r r_2$$

iff

$$\forall t_1 \in r_1, \forall t_2 \in r_2, \exists \sigma \in Sub, t_1 = \sigma t_2 \text{ or } t_1 = \bot_n,$$

where *Sub* is the set of all substitutions.

The relation $\sqsubseteq_\tau$ is a partial order and n_TUPLES° is a complete lattice with $\sqsubseteq_\tau$ [Yoshida et al. 85]. For $\tau_1, \tau_2 \in n\_TUPLES^\circ$, the greatest lower bound of τ_1 and τ_2 is written $\tau_1 \sqcap_\tau \tau_2$. It is equal to $C_\tau(mgu(t_1, t_2))$ for $t_1 \in \tau_1$ and $t_2 \in \tau_2$. Next we define the semantic domain of *n_T_Relations*.

Def.15 $n\_T\_RELATIONS$ is defined as follows.

$$\mathcal{T} \in n\_T\_RELATIONS$$

iff

$\mathcal{T}$ is a finite subset of n_TUPLES°, and $\forall \tau_1, \forall \tau_2 \in \mathcal{T}, \tau_1 \sqsubseteq_\tau \tau_2$ implies $\tau_1 = \tau_2$.

In other words, the elements of $n\_T\_RELATIONS$ are sets of maximal elements in some finite subset of n_TUPLES°.

Def.16 The binary relation $\sqsubseteq$ on $n\_T\_RELATIONS$ is defined as follows.

$$\mathcal{T}_1 \sqsubseteq \mathcal{T}_2 \text{ iff } \forall \tau_1 \in \mathcal{T}_1, \exists \tau_2 \in \mathcal{T}_2, \tau_1 \sqsubseteq_\tau \tau_2$$

Proposition.1 The relation $\sqsubseteq$ is a partial order on *n_T_Relation*.

Def.17 We define binary operations $\sqcap$ and $\sqcup$ on $n\_T\_RELATIONS$ as follows.

$$\mathcal{T}_1 \sqcap \mathcal{T}_2 \equiv \{\tau | \tau_1 \in \mathcal{T}_1, \tau_2 \in \mathcal{T}_2, \tau = \tau_1 \sqcap_\tau \tau_2, \forall \tau' \in \mathcal{T}_1 \sqcap \mathcal{T}_2, \tau \sqsubseteq_c \tau' \text{ implies } \tau = \tau'\}$$

$$\mathcal{T}_1 \sqcup \mathcal{T}_2 \equiv \{\tau | \tau \in \mathcal{T}_1 \cup \mathcal{T}_2, \forall \tau' \in \mathcal{T}_1 \sqcup \mathcal{T}_2, \tau \sqsubseteq_c \tau' \text{ implies } \tau = \tau'\}$$

$n\_T\_RELATIONS$ is closed under both $\sqcap$ and $\sqcup$ operations. Next, we'll show that $\sqcap$ is the greatest lower bound operation and $\sqcup$ is the least upper bound operation over the partial order $\sqsubseteq$.

Lemma.1 For any $\mathcal{T}_1$, $\mathcal{T}_2 \in n\_T\_RELATIONS$,

$$\mathcal{T}_1 \sqcap \mathcal{T}_2 \sqsubseteq \mathcal{T}_1 \text{ and } \mathcal{T}_1 \sqcap \mathcal{T}_2 \sqsubseteq \mathcal{T}_2 .$$

Lemma.2 For any $\mathcal{T}$, $\mathcal{T}_1$, $\mathcal{T}_2 \in n\_T\_RELATIONS$,

$$\text{If } \mathcal{T} \sqsubseteq \mathcal{T}_1 \text{ and } \mathcal{T} \sqsubseteq \mathcal{T}_2, \text{ then } \mathcal{T} \sqsubseteq \mathcal{T}_1 \sqcap \mathcal{T}_2 .$$

Proposition.2 $\mathcal{T}_1 \sqcap \mathcal{T}_2$ is the greatest lower bound of $\mathcal{T}_1$ and $\mathcal{T}_2$ over the partial order $\sqsubseteq$.

Lemma.3 For any T_1, $T_2 \in n\_T\_RELATIONS$,

$$T_1 \sqsubseteq T_1 \sqcup T_2 \text{ and } T_2 \sqsubseteq T_1 \sqcup T_2 .$$

Lemma.4 For any T_1, $T_2 \in n\_T\_RELATIONS$,

$$\text{If } T_1 \sqsubseteq T \text{ and } T_2 \sqsubseteq T, \text{then } T \sqsubseteq T_1 \sqcup T_2 .$$

Proposition.3 For all T_1, $T_2 \in n\_T\_RELATIONS$, $T_1 \sqcup T_2$ is the least upper bound of T_1 and T_2 over the partial order $\sqsubseteq$.

From Proposition.2 and Proposition.3 it is easy to establish the conclusion.

Theorem The domain $n\_T\_RELATIONS$ is a lattice with the partial order $\sqsubseteq$.

4. Semantics of RBU operations

In this section, RBU operations are characterized as monotone functions from $n\_T\_RELATIONS$ to $m\_T\_RELATIONS$, or compositions of greatest lower bounds operation. First we introduce $PROJECT_p$ functions from $n\_T\_RELATIONS$ to $m\_T\_RELATIONS$ corresponding to $project_p$:$n\_T\_Relations \rightarrow m\_T\_Relations$ for a given extended permutation p. Then the functions corresponding to unification_restriction and unification_join are defined.

Def.18 For given $p \in e\_perm_{n,m}$ a mapping $PROJECT_p : n\_T\_RELATIONS \rightarrow m\_T\_RELATIONS$ is defined as follows.

$$PROJECT_p(T) = \begin{cases} \emptyset & \text{if } T = \emptyset \\ \{\perp_m\} & \text{if } T = \{\perp_n\} \\ \{r | r_1 \in T, t \in r_1, r = C_r(r_p(t))\} & \text{otherwise} \end{cases}$$

Proposition.4 For any $p \in e\_perm_{n,m}$, and T_1, $T_2 \in n\_T\_RELATIONS$, if $T_1 \sqsubseteq T_2$ then

$$PROJECT_p(T_1) \sqsubseteq PROJECT_p(T_2).$$

Functions corresponding to unification_restriction and unification_join are defined as follows.

Def.19 (1) For $T_1 \in 1\_T\_RELATIONS$ and $T_2 \in n\_T\_RELATIONS$, $n \geq i \geq 1$,

$$U\_RESTRICTION_i(T_1, T_2) \equiv PROJECT_p(T_1) \sqcap T_2$$

where $p \in e\_perm_{1,n}, p(1) = i$.

(2) For $\mathcal{T}_1 \in n\_T\_RELATIONS$ and $\mathcal{T}_2 \in m\_T\_RELATIONS, n \geq i \geq 1,\ m \geq j \geq 1,$

$$U\_JOIN_{i,j}(\mathcal{T}_1, \mathcal{T}_2) \equiv PROJECT_p(\mathcal{T}_1) \sqcap PROJECT_q(\mathcal{T}_2).$$

where $p \in e\_perm_{n,m+n-1}, p(k) = k,$

$$q \in e\_perm_{m,m+n-1},$$

$$q(l) = \begin{cases} n+l & \text{if } l < j, \\ i & \text{if } l = i, \\ n+l-1 & \text{if } l > j. \end{cases}$$

Example.2 Let $T_1, T_3 \in 2\_T\_Relations$, and $T_2 \in 1\_T\_Relations, p \in e\_perm_{2,1}$ be similar to Example.1. $\mathcal{T}_1$, $\mathcal{T}_3 \in 2\_T\_RELATIONS$, $\mathcal{T}_2 \in 1\_T\_RELATIONS$ corresponding to T_1, T_3 and T_2 are as follows.

$$\mathcal{T}_1 = \{C_\tau((f(x_1), a)), C_\tau((f(g(y_1)), z_1)), C_\tau((g(f(u_1)), a)), C_\tau((g(v_1), v_1))\},$$

$$\mathcal{T}_2 = \{C_\tau((g(f(b))))\},$$

$$\mathcal{T}_3 = \{C_\tau((a, f(x_2))), C_\tau((f(y_2), y_2))\}$$

Let $\mathcal{T}_{pjc} \in 1\_T\_RELATIONS$, $\mathcal{T}_{rst} \in 2\_T\_RELATIONS$, and $\mathcal{T}_{join} \in 3\_T\_RELATIONS$ be semantics for $project_p(T_1), u\_restriction_1(T_2, T_1)$ and $u\_join_{2,1}(T_1, T_3)$ respectively, then ,

$$\mathcal{T}_{pjc} = \{C_\tau((f(x_1))), C_\tau((g(v_1)))\}$$

$$\mathcal{T}_{rst} = \{C_\tau((g(f(b)), a)), C_\tau((g(f(b)), f(b)))\}$$

$$\mathcal{T}_{join} =$$

$$\{C_\tau((f(x_1), a, f(x_2))), C_\tau((g(a), a, f(x_2))), C_\tau((g(f(u_1)), a, f(x_2))$$
$$C_\tau((f(g(y_1)), f(y_2), y_2)), C_\tau((g(f(y_2)), f(y_2), y_2))\}.$$

We can show:

$$\mathcal{T}_{pjc} = PROJECT_p(\mathcal{T}_1)$$

$$\mathcal{T}_{rst} = PROJECT_q(\mathcal{T}_2) \sqcap \mathcal{T}_1$$

$$\mathcal{T}_{join} = PROJECT_{r1}(\mathcal{T}_1) \sqcap PROJECT_{r2}(\mathcal{T}_3)$$

where $q \in e\_perm_{1,2}, q(1) = i,$

and $r1 \in e\_perm_{2,3}, r1(k) = k,$

$$r2 \in e\_perm_{2,3}, r2(1) = 2, r2(2) = 3.$$

5. Summary

A mathematical foundation for the semantics of a relational knowledge base was described in this paper. We excluded discussion of the domains of each term in term relations. In this project we plan to develop a knowledge base system that handles a variety of knowledge objects uniformly. The knowledge handling programs and the constraints on the knowledge in the system are also stored in a meta-knowledge base in the form of term relations. Each term in the meta-knowledge relations would be RBU expression. Thus, the domain of these terms is $n\_T\_RELATIONS \rightarrow m\_T\_RELATIONS$. When discussing the semantics of the knowledge base, this correspondance between the domain of terms and the domain operations must be considered.

ACKNOWLEDGMENTS

The authors thank Dr. M. Yoshida of Kyoto University and Mr. K. Sakai, Mr. K. Yokota, Mr. S. Ohyagi and Mr. A. Ohsuga of ICOT research center for many useful discussions.

REFERENCES

[Kakuta, et al. 85] Kakuta, T., Miyazaki, N., Shibayama, S., Yokota, H., and Murakami, K. The Design and Implementation of Relational Database Machine Delta, *Proceedings of the International Workshop on Database Machines '85*, March 1985.

[Scott 77] Scott, D. Logic and Programming Languages, *CACM,Vol.20.* 1977.

[Yokota, et al. 85] Yokota, H., and Itoh, H. A Model and an Architecture for a Relational Knowledge Base, *ICOT Technical Report No. TR-144*, submitted to The 13th International Symposium on Computer Architecture, Nov. 1985.

[Yoshida, et al. 85] Yoshida, M., Doshita, S. and Yamasaki, S. Dousyutsu Genri ni Okeru Chikan no Sokuron ni Motozuku Teisikika, (Formalization of Substitution in Resolution Principle with Lattice Theory), *Technical Report of IECE,AL85-5,pp35-41*, May. 1985, in Japanese.