

THEORETICAL BASIS FOR A KNOWLEDGE REPRESENTATION SYSTEM

Setsuo Ohsuga
Institute of Space and Aeronautical Science
University of Tokyo
4-6-1 Komaba Meguro-ku Tokyo 153
Japan

Theoretical basis for a system named the SBDS (Structure Based Deduction System) including a formalism for representation of knowledge and an inferential algorithm is presented. This system is designed as a part of an intelligent interactive system to support man carrying out such the intelligent activities as engineering design, decision making, scientific research and so forth. The major emphases on the system are : (1) high processing efficiency, (2) high expressive power so that user can represent any knowledge he wishes and (3) easy access to databases.

1. INTRODUCTION

A system named the SBDS (Structure Based Deduction System) including a formalism for representing knowledge and an inference algorithm is presented. It is a subsystem of a total knowledge system named the KAU (Knowledge Acquisition and Utilization) system that is designed as an intelligent interactive system intended to support man's intelligent activities such as engineering design, decision making, scientific research and so forth. Thus the motivation for the research resides in what is called the intelligent agent view point by E. Feigenbaum [6].

SBDS adopts, as its basic framework for representation of knowledge, an extended form of the first order predicate logic, because it is, we think, suitable for converting to and/or being converted from any of three major forms of information used in computers today, i.e., (1) languages, (2) databases and (3) programs. In this paper, relations of (1) and (2) to SBDS are discussed. That of (3) to the first order logic has been discussed in [5] and [10].

As a matter of course, a knowledge system is required to have high expressive power in representation of knowledge. Though there have been some approaches adopting the first order logic in AI problems [5], [7], [10] and also in the more practical problem of databases [3],[9], the first order logic shows a shortage of expressive power for many real applications. The adoption of the higher order logic in its

general form [8] as the main framework instead of the first order logic, however, is not good solution because it is too a general form for the practical purpose and therefore the inference is inefficient to be used in a real environment.

We therefore adopt a simpler form that is obtained by defining, first, a formalism to define the predicate and a deductive inference algorithm and, then, extending the first order logic towards higher order within the framework until the extended form can accommodate most of the language expressions that are almost sufficient to represent knowledge in many applications. The formalism for defining the predicate is combined with the hierarchical structure of entities defined in the universe. A fast deductive algorithm has been developed so as to use effectively the property of the structure.

On the other hand, for users working in the real applications, the use of databases is becoming inevitable. So far a number of databases and their management systems have been developed [4]. These DBMS's, however, are very inconvenient to be used for such the intelligent works as mentioned above. For an engineering designer, for example, the objective is to reach a final model through the repetitive processes of a model construction and its evaluation. He will require the data structure to be dynamically managed. But the conventional DBMSs can hardly do so because in these systems the structures of data are almost

fixedly defined. Therefore it is desirable that the knowledge system takes over the dynamic restructuring of information and can access to databases only when the values of data are necessary. SBDS possesses such the capability. Thus the KAU of which the SBDS occupies the central part is intended to serve as an intelligent interface between users and DBMS's [12], [13].

A number of different approaches have been made and have also been developing for the problem of knowledge representation [1], [15]. The approach proposed in this paper is still different from them as seen from the system's point of view though it implies some concepts common to the others in part. Theoretical basis of our approach is on the first order predicate and the resolution principle [2], [14]. However as the SBDS also contains the concept of network, there are also some ideas in common between our method of deductive inference and methods developed for performing deductive inference on the knowledge represented in the network [15].

In this paper, we describe the SBDS with its mathematical foundation. A universe is defined as the collection of all concepts that can be referred through the language. Over this universe, the first order logic is defined and then extended toward higher order to the limited extent. A classical predicate logic is used rather than the resolution logic not only because it is intuitively understood by many people but, rather, because it is convenient to extend the first order logic and also to convert the predicates to the database operations.

An inference algorithm has been developed in which the information contained in user's query is effectively used for searching information and for deduction. The algorithm is also applicable to the extended form by slightly modifying its control mechanism.

2. INTERFACE TO LANGUAGE FOR EXPRESSING KNOWLEDGE

A problem world is described in many cases by means of language, and, therefore, we consider only the world that is describable by the language.

In reality, a vocabulary allowed to be used in the knowledge system is finite in which some words, -nouns- designate sets of entities or individual entities while the order words -verbs, adjectives, propositions, some nouns,

etc.- represent the attributes of entities or the relations amongst entities. We call the former the entity words and the latter the relational words.

The syntax of the language defines sentences. There are a set of basic patterns of sentences or basic sentences in short such that all sentences can be analysed to the combination of some of them. If a system can understand the meaning of each basic sentence and also is given a parsing algorithm for analysing the compound / complex sentence to the logically connected set of basic sentences, then the system can understand the language. Here, by the term "understand" we mean that the system can process the information given in the form of language based on its meaning so that it responds correctly to users requests.

The set of basic patterns of sentences is finite in ordinary languages. In this paper, twenty-five patterns as shown in Table 1 are considered almost enough in case of English for expressing knowledge used in most of applications.

The vocabulary, on the other hand, is different in every application. Though some common words can be defined independent of applications, some jargons are to be added in each application. Therefore the dictionary in the system must be an open set.

3. DATABASE

A relational model is considered exclusively in this paper [3], [4] though the system is not restricted to it. In this model a relation is defined as a set of n-tuple records, $(t_1, t_2, \dots, t_n)$, where t_i is an occurrence value in the domain T_i . A relation is represented in the form of a flat table with n columns and a relation name is given to it.

4. FIRST ORDER PREDICATE LOGIC IN A CLASSICAL FORM

In the classical predicate logic, an atomic formula or atom in short, is defined as $F(t_1, t_2, \dots, t_n)$ where F is an n-place predicate symbol and $t_1, \dots, t_{n-1}$ and t_n are terms. Then a well formed formula, or a formula in short is defined recursively as follows [2] :

- a. An atom is a formula.
- b. If F and G are formula, then $\sim F$, $F \cup G$, $F \cap G$, $F \Rightarrow G$ and $F \Leftrightarrow G$ are formulas where $\sim$, $\cup$, $\cap$, $\Rightarrow$ and $\Leftrightarrow$ are read as "negation", "disjunction",

"conjunction", "imply" and "equivalent" respectively.

c. If F is a formula and x is a free variable in F then $(\forall x)F$ and $(\exists x)F$ are formulas.

d. Formulas are generated only by a finite number of applications of a through c.

A formula can be written in the prenex normal form. In the following, it is written in a vector form as :

$(Qx)[S\{F_i(x_i)\}]$, - - - (1)
where $(Qx) = (Q_1x_1)(Q_2x_2) \dots (Q_nx_n)$ with each Q_i being either $\forall$ or $\exists$, $S\{ \}$ stands for a logical structure composed of atoms, $F_i, (i=1 \dots m)$, by means of logical connectives shown in the above definition of the formula, and $(x_i) = (x_1 \dots x_{ini})$.

In the resolution logic [2],[4], the formula is standardized by being converted to the conjunctive normal form and Skolemized. In this paper, instead of the standardization, the formula is modified in another way.

Firstly, no function symbol appears in the formula. In general, n -place function is a mapping from D^n to D where D denotes a domain of a variable. It is represented in the SBDS by an $n+1$ -place atom $*F_f(z; x_1 \dots x_n)$ that is defined true iff $f(x_1 \dots x_n) = z$ holds. Since to each of this kind of atoms corresponds a procedure that evaluates it procedurally, these atoms are called the procedural type atoms. The procedural type atom is identified by a symbol $*$ on the left shoulder in the paper. An example is, $*(ADD\ 5; 2\ 3): "2\ plus\ 3\ is\ 5"$.

Secondly, in the resolution logic, Herbrand universe is defined as a set of which the elements are generated successively by means of the constants and function symbols appearing in the formula(s) while, in the SBDS, the universe U is first defined as the set of all objects that can be referred by the language.

Thirdly, the concept of sets is explicitly included in a formula. In the following, a variable free formula is called a ground formula. Any subset I of the set of all ground atoms, $F(t_1, \dots, t_n)$, of which $t_1, \dots, t_n$ are in the universe determines an interpretation where the meaning of an n -place predicate symbol F is a n -ary relation over the universe: $\{(t_1, \dots, t_m): F(t_1, \dots, t_n) \in I\}$. I is said to be a model of the formula if the truthhood of the formula is determined by the rules: F is true iff $F \in I$ and $\sim F$ is true iff $F \notin I$. When

I is a model, the set of the tuple $\{(t_1, \dots, t_n) | F(t_1, \dots, t_n) \in I\}$ is denoted D_I . Then $F(x_1, \dots, x_n)$ is true iff $(x_1, \dots, x_n) \in D_I$. This fact is written as
 $(\forall x)[(D_I \ni x) \Rightarrow F(x)]$ where $x = (x_1, \dots, x_n) \dots (2)$

In many cases, D_I is a Cartesian product of $X_i, (i=1, 2, \dots, n)$, i.e., $D_I = X_1 \times \dots \times X_n$ where each X_i denotes a domain of the variable x_i . In this case, the formula (2) is rewritten as
 $(\forall x_1) \dots (\forall x_n)[(X_1 \ni x_1) \cap \dots \cap (X_n \ni x_n) \Rightarrow F(x_1, \dots, x_n)]$ - - - (3)

or, equivalently,
 $(\forall x_1) \dots (\forall x_n)[X_1^*(x_1) \cap \dots \cap X_n^*(x_n) \Rightarrow F(x_1, \dots, x_n)]$ - - - (4)

where $X_i^*(x_i)$ is a predicate meaning " $X_i \ni x_i$ ". On the other hand, a fact that $F(x_1, \dots, x_n)$ is true for some value $(t_1, \dots, t_n)$ in D_I is written as

$(\exists x)[(D_I \ni x) \cap F(x)]$. - - - (5)

If D_I is a product set, then
 $(\exists x_1) \dots (\exists x_n)[X_1^*(x_1) \cap \dots \cap X_n^*(x_n) \cap F(x_1, \dots, x_n)]$. - - - (6)

If both $\forall$ and $\exists$ appear in a formula, then a recursive relation is obtained:
 $(Q_1x_1) \dots (Q_nx_n)[(X_1^*(x_1) \overset{?}{\cap} (X_2^*(x_2) \overset{?}{\cap} \dots \cap X_{n-1}^*(x_{n-1}) \overset{?}{\cap} (X_n^*(x_n) \overset{?}{\cap} F(x_1, \dots, x_n)) \dots)]$ - - - (7)

where $\overset{?}{\cap}$ represents either $\Rightarrow$ or $\cap$ depending on Q_i being $\forall$ or $\exists$. For example, if $D_I = X \times Y \times Z$, $Q_x = \forall, Q_y = \exists$ and $Q_z = \forall$, then
 $(\forall x)(\exists y)(\forall z)[X^*(x) \Rightarrow (Y^*(y) \cap (Z^*(z) \Rightarrow F(x, y, z)))]$ - - - (8)

In these representation of formulas, each variable is defined over U of which $X_i, (i=1, \dots, n)$, is a subset. Then the formula can also be written in an abbreviated form as

$(Q_1x_1/X_1) \dots (Q_nx_n/X_n) F(x_1, \dots, x_n)$ - - - (9)

meaning that the formula is true for assignment to each variable x_i of all or some values of X_i depending on the quantifier Q_i being $\forall$ or $\exists$, but otherwise it is not true. It is further simplified by abbreviating the variable symbols and moving F into the parenthesis as

$(Q_1X_1) \dots (Q_nX_n)(F, X_1, \dots, X_n)$ - - - (10)

With this formalism, (2) and (5) are written as $(\forall D_I)(F D_I)$ and $(\exists D_I)(F D_I)$ respectively. A ground atom $F(t_1, \dots, t_n)$ is a reduced form of (9) in which every domain has only one element. It should be noted that the formula (10) can be transformed easily to a set-of-clauses form of the resolution logic.

Example:

$(\forall X)(\exists Y)(\forall Z)(F X, Y, Z) = (\forall x)(\exists y)(\forall z)[(\sim X^*(x) \cup (Y^*(y) \cap (\sim Z^*(z) \cup F(x, y, z)))] = (\sim X^*(x) \cup Y^*(f(x)) \cap (\sim X^*(x) \cup \sim Z^*(z) \cup F(x, f(x), z)) = \{C_1, C_2\}$
 $C_1 = \sim X^*(x), Y^*(f(x)),$
 $C_2 = \sim X^*(x), \sim Z^*(z), F(x, f(x), z)$ - - - (11)
where $f(x)$ is the Skolem function.

The general form of the formula, then, is
 $(Q \ X)[S' \{(F_i \ X)\}]$ - - - (12)

5. THE UNIVERSE AS AXIOMATICALLY DEFINED SET

A predicate symbol of a formula corresponds to a relational word in a vocabulary set. Let the simple collection of all predicate symbols be R . Then since the vocabulary is a finite set, R is also finite. Each F_i in R acts on U to define a basic pattern of atomic formula of which the predicate symbol is F_i . In other words, to each F_i , an ordered set $\{t_1, \dots, t_{ni}\}$ is specified where each t_i is in U such that the ground atom $(F_i \ t_1, \dots, t_{ni})$ is in a model I .

The universe U is not so simple as R . Many different types of elements must be included in U . Consider, for example, two atomic formulas; (FATHER t_1, t_2): " t_1 is father of t_2 " and (CARD-NO t_3, t_4): "cardinal number of t_3 is t_4 ". Both t_1 and t_2 are of the same type in a sense that they can be the elements of a set, say, PERSON. On the other hand, t_3 is a set while t_4 must be an integer. In this case t_3 is said being of the different type and is one level higher than t_4 . Each t_i in an atom can be of the different level as was seen in the case of CARD-NO.

These examples shows that any set of the elements in U must be an element of U too. Each predicate symbol specifies the level of each term to be an argument of the ground atom. Such a term is called a ground term for the atom and a symbol # is attached in the following.

On the other hand, in the atomic formula (10) defined as the collection of all ground atoms, X_i is a set of t_i 's, and therefore is an element of U that is one level higher than t_i 's. Thus an atom and its ground atom are defined over two levels of elements in U for each argument.

This definitions of formula requires the universe U being of multi-layers structure. For example, the concept of the power set is necessary to represent a formula such as : (VT₃)($\exists$ INTEGER)(CARD-NO T_3 INTEGER): "For every set is there an integer which is the cardinal number?", where T_3 is a set of all sets. Fortunately such the set has been studied as an axiomatic set theory. We can use the Zermelo/Frankel axioms of set theory [11] as the universe though we don't touch on them here. We only point out that the set thus defined satisfies all conditions mentioned above U and, still further, such the set contains n -tuples, $(t_1, \dots, t_n)$ too, as it elements.

Based on this structure of the universe, a sim-

ple formalism for defining formula is obtained. That is, a formula is a pair $(F_i \ D_i)$ attached by the information for quantification, Q where $F_i \in R$ and D_i is a mapping of F_i into U .

6. INFERENCE ALGORITHM

Inference algorithm is defined to the formula in the form of (12). This algorithm is logically equivalent to the resolution principle [2] but is designed to have high physical performance. It consists of the selection rule (SR), the test for implicative condition (TIC) and the replacement rule (RR). TIC corresponds to the unification algorithm and RR the resolution in the resolution logic respectively.

Suppose a formula P of the form
 $P : (Q_p X)[(B_1, X_1) \cap \dots \cap (B_k, X_k) \Rightarrow (A, X_0)]$ - - - (13)
exists in the knowledge base where $(Q_p X) = (Q_{p1} X_1) \dots (Q_{pm} X_m)$, $X_i = \{X_{i1}, \dots, X_{i\alpha_i}\}$ and $X = \{X_1, \dots, X_m\} = \bigcup_{i=1}^m X_i$, and a query Q of the form
 $Q : (Q_q Y)[(C_1, Y_1) \cap \dots \cap (C_j, Y_j)]$ - - - (14)
is presented where $(Q_q Y) = (Q_{q1} Y_1) \dots (Q_{qn} Y_n)$, $Y_j = \{Y_{j1}, \dots, Y_{j\beta_j}\}$ and $Y = \{Y_1, \dots, Y_n\} = \bigcup_{j=1}^n Y_j$.

Then a non-procedural type literal C_j is selected from Q by SR and the relation
 $(Q'_p X_0)(A, X_0) \Rightarrow (Q'_q Y_j)(C_j, Y_j)$ - - - (15)
is tested by TIC where $(Q'_p X_0)$ and $(Q'_q Y_j)$ are the partial sequences of $(Q_p X)$ and $(Q_q Y)$ comprising only the variables included in X_0 and Y_j respectively. Implicative condition (15) is tested by the following two steps:

- (A) For each pair of corresponding variables, say X_{pi} and Y_{qi} , of left hand term and right hand term of (15), either of the set theoretical relations shown in Table 2 should hold depending on the combination of quantifiers of variables. This rule applies to all variables in the atoms of (15).
- (B) There should not be such indices pair (i, j) that, in $(Q'_p X_0)(A, X_0)$, i -th variable is universally quantified and is preceding to j -th existentially quantified variable while, in $(Q'_q Y_j)(C_j, Y_j)$, corresponding i -th universally quantified variable is preceded by j -th existentially quantified variable.

If the implicative relation (15) is certified by TIC, then a new formula R of the form
 $R : (Q_r Z)[(C_1, Z_1) \cap \dots \cap (C_{j-1}, Z_{j-1}) \cap (B_1, W_1) \cap \dots \cap (B_k, W_k) \cap (C_{j+1}, Z_{j+1}) \cap \dots \cap (C_l, Z_l)]$ - - - (16)
is generated by RR where $(Q_r Z) = (Q_{r1} Z_1) \dots (Q_{ri} Z_i) \dots (Q_{rn} Z_n)$, $Z_j = \{Z_{j1}, \dots, Z_{j\beta_j}\}$, $W_i = \{W_{i1}, \dots, W_{i\alpha_i}\}$, and $Z = \{Z_1, \dots, Z_n\} = \bigcup_{i=1}^n Z_i \vee \bigcup_{i=1}^m W_i$. Each Z_i and its quantifier Q_i is decided by its corresponding variables, X_i and Y_i , in P and Q .

respectively, and their quantifiers as shown in Table 3.

Q_{pi}	$Q_{qi}(\overline{Q}_{qi})$	CONDITION
$\forall$	$\forall(\exists)$	$X_i \supset Y_i$
$\forall$	$\exists(\forall)$	$X_i \wedge Y_i \neq \phi$
$\exists$	$\exists(\forall)$	$X_i \subset Y_i$

Table 2 Implicative condition between single 1-atom formula

	Q_{pi}	Q_{qi}	Q_{ri}	Z_{ri}/W_{ri}
$X_i \in (X_0 = Y_j)$	$\forall$	$\forall$	$\forall$	Y_i
	$\forall$	$\exists$	$\exists$	$X_i \wedge Y_i$
	$\exists$	$\exists$	$\forall$	X_i
$Y_i \in (Y - Y_j)$	-	$\forall$	$\forall$	Y_i
	-	$\exists$	$\exists$	Y_i
$X_i \in (X - X_0)$	$\exists$	-	$\forall$	X_i
	$\forall$	-	$\exists$	X_i

Table 3 Replacement rule

It is not difficult to show that above rules of TIC are equivalent to the condition for the unification algorithm to terminate successfully in the resolution logic.

The set theoretical relations appearing in Table 2 can be represented in the logical form as

$$(X \supset Y) \equiv (\forall x) [Y^*(x) \Rightarrow X^*(x)] \quad - - - (17)$$

$$(X \wedge Y \neq \phi) \equiv (\exists z) [X^*(z) \wedge Y^*(z)] \quad - - - (18)$$

The relation (15) is proved by showing that the negation of the relation, i.e.,

$$\sim[(Q'_p X_0)(A X_0) \Rightarrow (Q'_q Y_j)(C_j Y_j)] \equiv (Q'_p X_0)(A X_0) \wedge (\overline{Q}'_q Y) \sim (C_j Y) \quad - - - (19)$$

does not hold where $\overline{Q}_q$ is obtained by exchanging $\forall$ and $\exists$ of each Q_{qi} in Q'_q . (20) is transformed to the set-of-clauses form. Let it be denoted S. S is different to each assignment of quantifiers to variables. To avoid the notation becoming too complicated, let's pay an attention for a particular pair of variables (X_i, Y_i) . The quantifiers of (X_i, Y_i) is denoted $(Q_{pi}, \overline{Q}_{qi})$. Then four different forms of S are derived according to the different speci-

fication for $(Q_{pi}, \overline{Q}_{qi})$. Then the validity of conditions listed in Table 2 is tested by applying a refutation process to these conditions combined with S. Here, only a case, $(Q_{pi}, \overline{Q}_{qi}) = (\forall, \exists)$, is shown.

In this case, $S = \{\dots, \sim X^*_i(x_i), \dots, A(\dots x_i \dots)\}$, $\{\dots Y^*_i(f_{yi}(\dots))\}$, $\{\sim C_j(\dots f_{yi}(\dots))\}$ where $\dots$ represent terms concerning to variables other than x_i and y_i . $f(\dots)$ is a Skölem function where $\dots$ represents a set of universally quantified variables preceding to y_i in the prefix $(Q_q Y_j)$.

These set of clauses are put together with (17) in the standard form, $\{\sim Y^*(x), X^*(x)\}$. x_i can be substituted by $f_{yi}(\dots)$ because by the second law of TIC it is ensured that $\dots$ does not include x_i . Then $A(\dots x_i \dots)$ and $C(\dots f_{yi}(\dots))$ is unifiable if A and C is the same. Thus it is easy to show that the empty clause is deduced from the set of these clauses. The other cases are also proved by the similar procedures.

Example :

P: $(\forall X)(\forall Y)(\forall Z)[(\text{BROTHER } X \ Y) \wedge (\text{PARENT } Y \ Z) \Rightarrow (\text{UNCLE } X \ Z)]$ "For every x, y, z, x who is brother of z's parent y is z's uncle"
Q: $(\exists U)[(\text{UNCLE } U \ \# \text{Tom}) \wedge (\text{ILL } U)]?$ "Is there any uncle u of Tom, who is ill?"
where X, Y, Z and U represent the set PERSON and $x \in X$, $y \in Y$, $z \in Z$ and $u \in U$.

At the deduction SR selects $(\exists U)(\text{UNCLE } U \ \# \text{Tom})$ from Q and TIC tests for
 $(\forall X)(\forall Z)(\text{UNCLE } X \ Z) \Rightarrow (\exists U)(\text{UNCLE } U \ \# \text{Tom}) - - (20)$
It holds because $X \wedge U \neq \phi$, $Z \supset \{\# \text{Tom}\}$. Then RR generate,
R: $(\exists U)(\exists Y)[(\text{BROTHER } U \ Y) \wedge (\text{PARENT } Y \ \# \text{Tom}) \wedge (\text{ILL } U)] - - (21)$
R is regarded as a new query.

The deductive procedure is repeated until either the formula of which the Boolean value can be obtained or the formula consisting only of the procedural type literals is reached. Further consideration is necessary to make the algorithm logically complete though it is abbreviated here. Refer [13] for the detail.

7. STRATEGY FOR SEARCHING THE CANDIDATE FORMULA

There may be a number of formulas stored in a knowledge system and, at a time only part of which participate to deducing the query. Therefore, quick selection of the relevant formula is very important.

The universe is represented by a hierarchical structure in which each node represents a set of objects. In the following a set is denoted by the upper case letter. A node is connected to another node by either of two types of arcs representing the set-theoretical inclusion and disjointness. The former is called the I-type arc and the latter the D-type arc. For instance, the set LIVING-THING (LIVTH in short), MAMMALIA, FISHES, BIRDS, MANKIND, DOGS, CATS etc are in the set-theoretical relations such as $LIVTH \supset MAMMALIA$, $FISH$, $BIRD$, $INSECT$, ... $MAMMALIA \supset MANKIND$, DOG , ... and so on. We say "X is upper to Y" or "X is larger than Y" when $X \supset Y$ and Y is connected to X by the I-type arc. On the other hand, the set MAN and WOMAN are disjoint to each other and are connected by the D-type arc. By applying the processes of linking by means of these arcs successively, a hierarchical structure of universe, called the semantic skeleton structure, or the skeleton in short, is formed. Some node is connected to more than two other upper nodes by the I-type arcs. Such a node represents a set defined as an intersection of the other sets. For instance, a set BOY is represented as the intersection of MAN and CHILD.

A formula that includes a set X_i as an argument is linked to the node X_i in the skeleton and is accessible from this node. Accordingly the formula (10) can be accessed from any of nodes $X_1, X_2, \dots, X_n$.

Suppose a query (14) includes a constant or a variable, say Y_j , that is universally quantified. It is ordinary the case because a query of which every argument is existentially quantified variable is nothing else but a damp command. Then the formula P that satisfies the relation (15) to this query, if any, must be linked to some node X_j that is over Y_j in the skeleton from the rule (A) of TIC. Therefore, it is reachable by following the I-type arcs upward starting from Y_j . Since the skeleton expands downward, it reduces the search space considerably and formulas of which the predicate symbols are the same with the query but the variables are out of above condition are disregarded.

The formula thus found is called the candidate which satisfies the condition of TIC only with respect to the variable Y_j . It must be tested for the condition (Table 2) for the other variables too. The skeleton is used again. Since, this time, both left and right hand formulas of (15) are in hand and, therefore, both domains (sets) of corresponding arguments can be specified in the skeleton, the tests for the set theoretical relations of Table 2 are achieved

by the test for the positional relations between nodes in the skeleton. Since its algorithm is very simple it can be implemented by the special but simple wired logic and therefore the test can be performed very fast. The SR selects a literal among all literals in the query that contains most constants / universally quantified variables.

The system further reduces the possibility of fruitless test being achieved. In case of conventional first order logic, the set of formulas such as

$$B_1 \Rightarrow A, B_2 \Rightarrow A, \dots, B_e \Rightarrow A \quad - - - (22)$$

often arises in the definition of the meaning of words. For instance, the relations between sets mentioned above are represented as

$$\left. \begin{aligned} (\forall x) [MAMMALIA^*(x) \Rightarrow LIVTH^*(x)] \\ (\forall x) [FISH^*(x) \Rightarrow LIVTH^*(x)] \\ (\forall x) [BIRD^*(x) \Rightarrow LIVTH^*(x)] \end{aligned} \right\} \quad - - - (23)$$

Suppose $A \Rightarrow Q$ is satisfied to a query Q. Then the resolution logic will make an exhaustive test looking for a path leading to the empty clause. It is exhaustive because the refutation process proceeds in the direction from A to B_i in each formula (22) in a sense that A is unified with the query and B_i is remained for the next test.

In the SBDS every formula of the type (23) is represented by an arc in the skeleton. Searching the formula by following the I-type arcs upward in the skeleton corresponds to searching the refutation path proceeding in the reverse order of the resolution logic. Thus, as far as such the formulas as (23) are concerned, the exhaustive test is avoided.

8. ACCESS TO DATABASES

The formula (2) means that if a tuple $x = (x_1, \dots, x_m)$ is included in D, a relation F holds among $x_1, \dots, x_m$. Suppose then that a set of n-tuples, D, is given as a data file in a table form. Then, the condition (D)t for some n-tuple, $t = (t_1, \dots, t_m)$, is tested by a special procedure that searches t in the file. That is, the premise of (2) is represented by a kind of procedural type atom. Let it be denoted as $*(FGET, D : x_1, \dots, x_n)$ and be called the file type atom where D indicate the file name. Then the formula (2) is written as $(\forall x_1) \dots (\forall x_n) [*(FGET D : x_1, \dots, x_n) \Rightarrow F(x_1, \dots, x_n)]$ - - - (24)

This formula gives a definition/description of a table type file D. A user input the system a set of data in the table form accompanied with a formula (24). After then, any user can use this file without any knowledge on it. The

system determines itself how and when to access the file through the deductive operation and converts the formula to the relational algebra [4]. Only the outline of the conversion is shown without proof.

Suppose, a formula consisting only of the file type atoms.

$$(Q \ X) [S \{*(FGET \ D_i : X_i)\}] \quad - - - (25)$$

is obtained as the result of deductive operation, where $X_i = (X_{i1}, \dots, X_{ini})$ and $X = (X_1, \dots, X_n) = \bigvee_{i=1}^n X_i$. Suppose each file D_i is the set of m_i -tuples and each domain X_i is a finite set.

$$\text{Let } X_i^* = X - X_i. \text{ Then (25) is equivalent to} \\ (Q \ X) [S \{*(FGET \ D_i \times X_i^* : X)\}] = (Q \ X) [*(FGET, \hat{S} \{(D_i \times X_i^*)\}, X)] \quad - - - (26)$$

where $D_i \times X_i^*$ means a cylinder set in the n -dimensional space, $X_1 \times \dots \times X_n$, $\hat{S}$ is a structure of the set theoretical operations that is obtained from S by replacing every $\cap$ and $\cup$ in S by Δ (intersection) and $\vee$ (union) respectively, and $\hat{S} \{(D_i \times X_i^*)\}$ means that the structure of the set operations $\hat{S}$ are applied to $(D_i \times X_i^*)$. Since all the $(D_i \times X_i^*)$'s, $(i=1, 2, \dots, k)$, are the set of n -tuples, $\hat{S} \{(D_i \times X_i^*)\}$ is reduced to a set of n -tuples. Let it be denoted D^* . Then the formula (26) results in

$$(Q \ X) [*(FGET \ D^* : X)] \quad - - - (27)$$

Here the following two file operations are introduced.

$$\text{DIV}(D^*, X_i) = \{x = (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) \mid x \times X_i \subset D^*\} \quad - - - (28)$$

$$\text{PRJ}(D^*, X_i) = \{x = (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) \mid (x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n) \subset D^*\} \quad - - - (29)$$

These operations correspond to division and projection operations respectively of the relational algebra in the relational data base.

Then the following relations between logical view and set-theoretical view hold.

$$\left. \begin{aligned} (Q_1 X_1) \dots (Q_{n-1} X_{n-1}) (\forall X_n) [*(FGET \ D^* : X_1, \dots, X_n)] &= (Q_1 X_1) \dots (Q_{n-1} X_{n-1}) [*(FGET \ \text{DIV}(D^*, X_n) : X_1, \dots, X_{n-1})] \\ (Q_1 X_1) \dots (Q_{n-1} X_{n-1}) (\exists X_n) [*(FGET \ D^* : X_1, \dots, X_n)] &= (Q_1 X_1) \dots (Q_{n-1} X_{n-1}) [*(FGET \ \text{PRJ}(D^*, X_n) : X_1, \dots, X_{n-1})] \end{aligned} \right\} \quad (30)$$

These relations and operations are repeatedly applied to the formula (27) until finally $*(FGET \ T : \phi)$ is obtained where T is either 1 or 0 depending on if the operations can be continued to the end or these operations result in an empty file. If $T=1$, the original query formula proved true.

In practice, user must be able to designate some data to be output. Such the data must be saved during the above processes. When the other procedural type atoms besides the file type are included in a formula, a further consideration is necessary. For the details refer [13].

9. EXTENTION OF THE SYSTEM

9.1 Predicate Symbol as Variable

A hierarchical structure can be introduced in the set of predicate symbols. Consider, for example, two atoms, (BROTHER $X \ Y$) and (SIBLING $X \ Y$). When the same terms are substituted into the corresponding variables, then (Brother, t_1, t_2) $\Rightarrow$ (Sibling, t_1, t_2). In this case "Sibling" is said larger than "Brother".

On the other hand, (Natural, X) $\cap$ (Artificial, X) shows an inconsistency. The predicate symbols being in such relations are said disjoint to each other. Based on these relations, the skeleton is formed for the predicate symbols and the predicate symbol can be dealt with as if it were a variable because it can be processed by the same algorithm as the other variables at the deduction. Then the atom can be rewritten as

$$(Q \ X)(X), \quad X = (X_0, X_1, \dots, X_n) \quad - - - (31)$$

where the predicate symbol is denoted X_0 . With this modification, the system can accept the query which asks the value in X_0 , i.e., the predicate symbol. For example, a query $Q: (\exists \text{RELATION})(\text{RELATION} \ \# \text{Tom} \ \# \text{Mary})? : \text{"What is the relation holding between Tom and Mary?"}$ can be answered by using $P: (\text{FATHER} \ \# \text{Tom} \ \# \text{Mary})$ if the relation $\text{FATHER} \subset \text{RELATION}$ is given in advance as is the case of the other variables.

9.2 Recursive Form

Let the atom of the form (31) be called the n -atom when it comprises n terms including X_0 . In ordinary predicate, n is larger than or equal to two. Suppose it can be extended to the case $n=1$. There are only two 1-atoms of the form, $(\forall X)(X)$ and $(\exists X)(X)$ with meanings "for all $x \in X, x \in X$ " and "there is some x in X such that $x \in X$ ", respectively. Thus (X) shows the logical aspect of the set X . The applications of TIC rules to the 1-atoms reveals the rule (A) of TIC being just the implicative condition itself for two 1-atoms.

The SBDS has been described as including the concepts both of sets and of logic. It is for the purpose of taking advantage of the strength of each at the implementation. We can obtain the logical description of it by replacing X by (X) and the rule (A) of TIC by the implicative relation between 1-atoms. Then the universe is regarded as a collection of 1-atom formulas and the formula (31) is rewritten as the relation among n 1-atom formulas to which the rule (A) of TIC, rephrased as the followings, is applied: (A') To each pair of corresponding inner 1-atom formulas, the implicative condition should hold.

These concepts are extended. Assume that the universe can include not only 1-atoms but general formulas, by adding all possible formulas definable by R and U to U resulting in U_1 , then by adding all possible formulas definable by R and U_1 to U_1 resulting in U_2 and so on. For many applications the extended universe U_k with finite k may be enough. The mechanism for defining the atom (ref. Sec.5) is unchanged. Then the SBDS has atoms in recursive form as $((FML)_0, (FML)_1 \dots (FML)_n)$ - - - (34) where $(FML)_0$ denotes a 1-atom corresponding to the predicate symbol and $(FML)_i, (i=1, \dots, n)$, is a formula. The rule (A') is extended to : (A'') For each pair of corresponding inner formulas, the implicative condition should hold. The rule (A'') is applied recursively to the formulas in form of (34) at the deduction.

As the result of this extension the SBDS acquires a large power of expression. But the extension is based on the assumption that any 1-atom formula can be replaced by a general formula. Further study on its validity is necessary.

Example: $(\forall X)(\forall U)[(INTEND\ X\ (DO\ X\ U)) \cap (CAN\ X\ (DO\ X\ U))] \Rightarrow (DO\ X\ U)$: "Every one does thing if he intends to and can do it."

where X ; PERSON, U ; UNIVERSE,
 $(CAN\ X\ (DO\ X\ U))$: X can do U ,
 $(INTEND\ X\ (DO\ X\ U))$: X intends to do U .

10. CONCLUSION

An outline for the SBDS, a central part of a knowledge system (KAU) designed as an intelligent support system, has been presented. Mathematical foundation for the formalism for representing knowledge, the deductive algorithm and the interface with databases has been mainly described. The major emphases on the SBDS design are : (1) high processing efficiency, (2) high power of expression so that user can represent any knowledge he wishes and (3) easy access to databases. The core of SBDS has been implemented and is being tested for its usefulness.

REFERENCES

- [1] Bobrow, D.G. (chaired) "A Panel on Knowledge Representation." In Proc. 5th IJCAI, 1977, pp.983-992.
- [2] Chang, C.L. & Lee, R.C.T. "Symbol Logic and Mechanical Theorem Proving." Academic Press, 1973.
- [3] Codd, E.F. "Recent Investigations in Relational Data Base Systems", Proc. IFIP Congress, Aug. 1974.
- [4] Date C.J. "An Introduction to Database Sys-

tems", Addison-Wesley, 1975.

- [5] Emden Van "Programming with Resolution Logic", Machine Intelligence 8, pp266-299.
- [6] Feigenbaum E.A. "The Art of Artificial Intelligence, Themes and Case Studies of Knowledge Engineering", 5th IJCAI pp1014-1029.
- [7] Green C. "The Use of Theorem Proving Techniques in Question Answering System", Proc. 23rd Nat. Conf. ACM, 1968.
- [8] Huet G.P. "A Mechanization of Type Theory" 3rd IJCAI 1973. pp.139-146.
- [9] Kellog C.H. & Klahr P. "Adding a Deductive Capability to a Data Management System", Second VLDB Sept. 1976.
- [10] Kowalski R.A. "Predicate Logic as Programming Language", Proc. IFIP-74. pp569-574.
- [11] Krivine J. "Introduction to Axiomatic Set Theory", D.Reidel Pub. Co. 1971.
- [12] Ohsuga S. "Semantic Information Processing in Man-Machine Systems", Proc. 1977 IEEE Conference on Decision & Control, pp1351-1358, Dec. 1977.
- [13] Ohsuga S. "Towards Intelligent Interactive Systems", Proc. the IFIP W.G.5.2 Workshop Seillac II on Methodology of Interaction (to appear) North-Holland Pub. Co. 1979.
- [14] Robinson J.A. "A Machine Oriented Logic Based on the Resolution Principle", JACM Vol.12 Jan. 1965.
- [15] Shapiro S.C. "Path-Based and Node-Based Inference in Semantic Networks", TINLAP-2. Jan. 1978 pp219-225.

- 1) S-V-Direct Object 2) S-V-to-Infinitive
- 3) S-V-Noun-to-Infinitive 4) S-V-Noun-(to be)-Complement 5) S-V-Noun-Infinitive 6) S-V-Noun-Present Participle 7) S-V-Object-Adjective
- 8) S-V-Noun 9) S-V-Past Participle 10) S-V-Adjective 11) S-V-that clause 12) S-V-Noun-that clause 13) S-V-Conjunctive-to-Infinitive
- 14) S-V-Noun-Conjunctive-to-Infinitive 15) S-V-Conjunctive Clause 16) S-V-Noun-Conjunctive Clause 17) S-V-Gerund 18) S-V-Direct Object-Preposition-Prep.Object 19) S-V-Indirect Object-Direct Object 20) S-V-(for)-Complement 21) S-V 22) S-V-Predicative
- 23) S-V-Adverbial Adjunct 24) S-V-Preposition-Prepositional Object 25) S-V-to-Infinitive

- Examples: 1) X likes Y : (LIKE $X\ Y$)
 2) X intends to do... : (INTEND $X\ (DO\ X\ \dots)$)
 12) X tells Y that Z does... : (TELL $X\ Y\ (DO\ Z\ \dots)$)
 17) X enjoys doing Y .. : (ENJOY $X\ (DO\ X\ Y\ \dots)$)
 19) X gives $Y\ Z$: (GIVE $X\ Y\ Z$)

Table 1 Basic patterns of sentences