PREDICATE LOGIC INVOLVING DATA STRUCTURE AS A KNOWLEDGE REPRESENTATION LANGUAGE

Setsuo Ohsuga

Institute of Interdisciplinary Research
Faculty of Engineering,
The University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo 153, Japan

ABSTRACT

A modification of predicate logic, called multi layered logic, is discussed. It is designed as the knowledge representation language for describing the systems that uses engineering knowledge to support the system analysis and design. A multi level data structure is used to represent the complex systems in which an abstract entity at a level of abstraction is formed as the collection of the entities at the next lower level. Multi layered logic adapts to this structure of the system and meets the conditions for supporting analysis and design of complex systems.

I Introduction

In order to analyze and design large systems, a structured analysis and design methodology has to be employed. This involves the concept of multiple levels of abstraction where at each level of abstraction, the system is viewed composed of a network of entities. Abstract entities at the next higher level of abstraction is formed as the collection of these entities. At the highest level of abstraction, the system is a single entity while at the lowest level, the system is composed of primitive entities. Each primitive entity has some functionality which is specified, in advance, for each object system. Systems of more complex functionality are composed of them. Here by functionality, we include various concepts' other than the structural relationships, such as attribute, property, function etc. of an entity. When to define the abstract entity at a level of abstraction, the internal structure of the entity at the next lower level need not be known, but these entities can be thought of as primitives at that level of abstraction. Some entities at the same level are interconnected to each other to form the network of entities. Thus the specification of the structure of a system is recursive in nature.

A system can be specified either directly by specifying its structure or indirectly by specifying its behavior. Generally speaking, to analyze a system is to obtain the behavior of the system given its structure. Hence the structure of the system has to be specified in terms of the primitive entities before it can be analyzed. On the other hand, to design a system is to obtain the structure of the system given the behavior. In general, to find the complex structure to have the specified behavior is not an easy task. One needs to generate a tentative structure and examine it whether it has the specified behavior. The system can be specified by specifying the structure and the functionality of the primitive entities at the lowest level. But very often it is necessary to give the explicit description on (part of) functionality of an entity at an arbitrary level. Thus, each abstract entity has three parts: (1) the components, (2) the interconnection between the entities in the same level and (3) the description on functionality. These parts are represented in the system described below by the hierarchical structure for (1) in which each entity at any level of abstraction is represented by a node and each node in the collection at the lower level is linked to the upper node, by the table of connection for (2) and by the logical formula for (3).

In this paper, we will discuss a knowledge representation language designed to deal with this structure. Unlike many of the CAD lnguage involving the hardware description language [3,4], this language is designed to use engineering knowledge to support the automatic transformation between the structure of the system and its behavior.

This language is based on the predicate logic because it is suited for describing the interconnections between the entities in the table form and functionality of an abstract entity. The entity-component relation also can be represented by providing the system with an atomic formula of the form (COMPONENT x y) meaning that x is a component of y. However, then, the representation of the complex structure of the system will be composed of a number of instances of the atomic formula and synthesizing, restructuring and analyzing the structure should be achieved by means of the logical inference. It is an inefficient way of processing the structure. Because these processing can be formalized to a large extent by viewing the entity-components relation as the set-elements relation, an efficient procedure to deal with the structure can be defined without loss of generality. For this purpose we modify the syntax of predicate logic to restrict the scope of the quantifiers just to the referenced structure. We call this system multi layered logic (MLL in short).

II Expression in MLL

The definition of MLL is mostly the same with the ordinary first order logic except the scope of each variable is restricted to a specified set and also the set can be an abstract entity in the hierarchical structure mentioned in the last section viewing the collection of the lower level entities as the set.

There is the concept of predicate logic with the restricted domain of variables, named many sorted logic (MSL in short). Here, the formula $(\forall x)[(\mathrm{MAN}\ x) \Rightarrow (\mathrm{MORTAL}\ x)]$ in the ordinary first order logic is written as $(\forall x/\mathrm{man})(\mathrm{MORTAL}\ x)$ where the variable x is restricted to the set 'man'.

Now referring to the hierarchical structure of (upper half of) Fig.l, let's consider an expression "There is some polygon in the polyhedron H such that, for every edge line in this polygon, its length is #1". This is the case where the object being described is an edge line which is referred to not directly but indirectly through the known entity H via entities at the intermediate level (polygon).

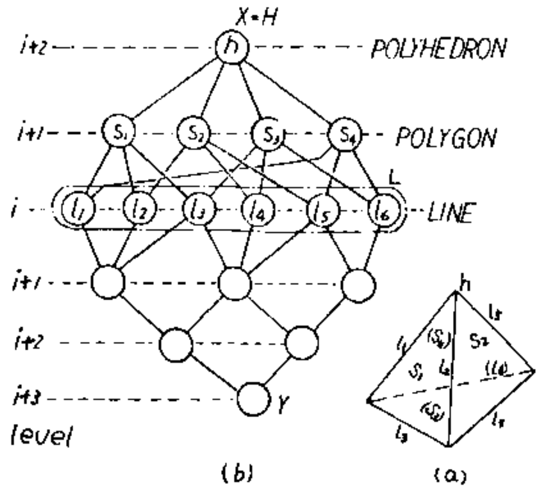

Fig.1 An example of hierarchical structure

Let's try to represent it by using MSL. "Some polygon in H" part is denoted $(\exists x/H)$, where x represents some polygon in H, while "for every edge line in this polygon" part, in the spirit of MSL, should be written as $(\forall y/x)$, resulting in the formula,

$(\exists x/H)(\forall y/x)$ [LENGTH it 1] ($1)

Note that x is a variable representing a polygon in H and at the same time a domain of another variable y (edge line). That is, the prefix of the formula reflects the given hierarchical structure. Thus MLL is the natural extension of MSL to allow an object being any abstract entity in the structure.

This idea is extended to the case where the structure is not specified but it is asked to find such a structure, given the set of primitive entities, to satisfy the specified condition. This is the case of automatic synthesis of the structure using the set of components L. In the syntax of multi layered logic, a symbol * is used to denote a power-set. Let *L denote the power-set of L (but the empty set is excluded). For example, if L is a set of lines, then *L can be interpreted as a set comprising all polygons composed of lines in L. Similarly, *(*L) defines a set that comprises all polyhedra composed of L. In general, n-th order power-set is defined recursively $*^n L = *(*^{n-1} L)$.

Suppose to ask the system to synthesize a polyhedron of the volume #b by using the given set of the lines L as the edge lines. Then the system should obtain at first the set S of polygons as the subset of *L, and then, a set of polyhedra V as the subset of *S which, in turn, is a subset of $*^2$L. The condition of an element x of *L being a polygon is that a chain of lines contained in x forms an elementary cycle and every line in this chain is on the same plane. Let the condition be represented (POLYGON x). Similarly, the condition of an entity y in $*^2$L being a polyhedron is that y satisfies Eular equation. Let the condition be represented (POLYHED y x). Then the requirement is represented,

$(\exists y/*^2 L)(\forall x/y)[(\mathrm{POLYHED}\ y\ x) \wedge (\mathrm{POLYGON}\ x) \wedge (\mathrm{VOLUME}\ y\ \#b)]$. ($2)

III Properties of Multi Layered Logic

Suppose, for example, two structures, say X and Y, are composed of the same set of primitive components, L, as shown in Fig.1. Then both

$(\forall x/X)(\forall y/x)(F\ y)$ ($3a)

$(\forall u/Y)(\forall v/u)(\forall w/v)(F\ w)$ ($3b)
are legal expressions and are equivalent to $(\forall z/L)(F\ z)$ which contains only the essential information involved therein. We call it the base expression. Then the system should obtain the base expression for the formula like ($3).

Assume that there is a common ground level and a level is assigned to each entity relative to this ground level. An i-th level entity is denoted x^i.

We call hereafter the quantifier-variable-domain triple, (Q x/X), QVD in short where Q represents either V or 3 . As shown in ($3), the representation of the prefix part becomes different even with the same matrix, depending on the structure referenced by the formula. Hence the set of the variables, say X, belonging to a structure and appearing in the prefix is not equel to the set X' of the variables in the same structure but appearing in the matrix. Let the highest level and the lowest level of the variables in X be n and k while those of the variables in X' be m and \ respectively. For example, in case of ($3b), X={u, v, w} X'= {w}, n=i+2, k=i, m=i+1, l=i. In general, n>m and 1>k. More than one groups of variables belonging to the different structures can appear in the same formula. The order of QVD in the prefix affects the meaning of the expression. In case of MLL, however, this order -meaning mapping is not one-to-one as is the case of ordinary predicate logic, because, for the variables belonging to the same structure, the order in the structure, from the top to the bottom, precedes the order in the prefix.

For example, compare $(\exists x^{i+1}/X)(\forall y/x^{i+1})$ (F y) with $(\forall y/x^{i+1})(\exists x^{i+1}/X)(F\ y)$. Both of them are defined in relation to the structure X of Fig.l. The orders of variables in the prefix and the structure referenced coincide in the first one while the orders differ in the second one. However, y can not be specified unless x^{l+l} is fixed, and the structural order acquires a priority. Then we interprete the second formula the same as the first one.

Suppose the formula is not in the base form. It is possible to modify the structure, by the following theorems, so that both the highest level and the lowest level of the variables involved in the prefix and in the matrix agree, still keeping the logical equivalence of the formulas*

Theorem : There is a base formual for any non-base formula.

Refer to [5] for the procedure to obtain the base formula. In the following, therefore, we consider only the base formula, that is, m=n and k=l.

IV Inference of MLL

We can obtain an inference rule for MLL as the modification of the resolution principle.

A. Standard Form

A formula of MLL is standardized by the same procedure as the ordinary case except a few differences as the followings. First, because the order in the prefix of variables belonging to the same structure has no effect, Skörem function replacing a variable comprises only variables belonging to the other structure as the arguments. Second, the domain specified to each variable should be remained. It is kept in the form of the variable-domain pair x/X denoting X is the domain of x.

Then the formula, for example,
$(\forall x^1/X)(\exists y^1/Y)(\forall y^0/y^1)(\exists x^0/x^1)$
$(F\ x^1\ x^0\ y^1\ y^0)$
is transformed to $(F\ x^1/X,\ f(y^0/g(x^1/X))/x^1,\ g(x^1/X)/Y,\ y^0/g(x^1/X))$.
Here x^0 is replaced by the function $f(y^0/y^1)$, but because y^1 is replaced by another function $g(x^1/X)$, it is substituted into y^1 in $f(y^0/y^1)$ resulting in $f(y^0/g(x^1/X))$.

B. Unification

Suppose two sets of clauses are given. A most general unifier has to be found between them. In case of MLL, not only the variables but their domains should be unified.

The pair of formulas are unifiable if and only if the condition shown in Table 1 is met on the set-theoretical relation between the domains of the highest level variables involved (denoted here X and Y), as well as the ordinary unifiability condition on terms (variables, functions and constants). Table 1 also shows the unified domain appearing in the resolvent. As an entity is the domain of the next

$(Q_x\ Q_y)$	Condition	New Domain
$(\forall\ \exists)$	$X \supset Y$	Y
$(\forall\ \forall)$	$X \cap Y \neq \emptyset$	$X \cap Y$
$(\exists\ \forall)$	$X \subset Y$	X

$(Q_x\ Q_y)$: combination of quantifires in both formulas.
X, Y : domains of variables

Table 1 Condition on domains for unifiability

lower level variables, that the uppper level variable is unified means that the domains of the next lower level variables are unified and the condition of Table 1 is automatically satisfied. Hence, no consideration on the domain is necessary at all for lower level variables in the same structure.

Example. A couple of formulas

$F_1 = (\exists x^1/X)(\forall y^1/Y)(\forall x^0/x^1)(\exists y^0/y^1)$
$\quad [\sim(G\ x^1, x^0, y^1, y^0) \vee (F\ x^1, x^0, y^1, y^0)]$
$F_2 = (\forall v^1/V)(\forall w^1/W)(\exists v^0/v^1)(\forall w^0/w^1)$
$\quad [\sim(F\ v^1, v^0, w^1, w^0)]$

are normalized, respectively, to

$\sim(G\ \ a^1/X,\ x^0/a^1,\ \ y^1/Y,\ \ f(x^0/a^1)/y^1),$
$\quad (F\ \ a^1/X,\ x^0/a^1,\ \ y^1/Y,\ \ f(x^0/a^1)/y^1)$
$\sim(F\ \ v^1/V,\ \ g(w^1/W)/v^1,\ \ w^1/W,\ \ w^0/w^1).$

The condition of unifiability on the domain is : $X \subset V$ and $Y \cap W \neq \emptyset$. Then the resolvent is

$\sim(G\ a^1/X,\ g(y^1/Z)/a^1,$
$\quad y^1/Z,\ \ f(g(y^1/Z)/a^1)/y^1)$

where $Z = Y \cap W$.

Before closing this section, we break further down the conditions presented in Table 1. It is necessary when either or both of X and Y is of the form $X = *^m A$, because not X but A is the known quantity and therefore the condition must be expressed in terms of A.

Let X^{m+1} and Y^{m+1} denote the highest level known entities and A^i and B^i be the i-th level known entities. Then Table 2 shows how the relations between the domains listed in Table 1 are broken down to those of known quantities. In this table, the symbol $\overset{r}{\downarrow} X^s$ denotes a set of entities at the level s-r and composing a specific structure X^S of the level s. The symbol '--' shows that the relation does not hold.

In Table 2, both non-specific structures X and Y at the level m+1 are assumed being composed of the i-th level entities. In general, however, it is possible that the levels of component sets differ, that is, $X = *^{m-i+1} C$ and $Y = *^{m-j+1} D$, $i \neq j$, where C and D are the known entities of the level i and j respectively. Without loss of generality, we assume $i < j$. Then the test for $X \supset Y$, $Y \supset X$ and $X \cap Y \neq \emptyset$ is further broken down to

(1) $X \supset Y = *^{j-i} C \supset D$ by the rule (I)-(1)
$\qquad = C \supset^{j-i}_{\downarrow} D$ by the rule (I)-(2)
(2) $X \subset Y = *^{j-i} C \subset D$ by the rule (I)-(1)
$\qquad$ does not hold by the rule (I)-(3)
(3) $(X \cap Y \neq \emptyset) = (*^{j-i} C \cap D \neq \emptyset)$
$\qquad$ by the rule (II)-(1)
$\qquad = C \supset^{j-i-1}_{\downarrow} D'$ by the rule (II)-(2)
$\qquad$ for some k

where $D = \{D_1^1, D_2^1, \ldots\}$
The new domains in the resolvent for the cases (1) and (3) are $*^{j-i} D$ and $*^{j-i}(\bigcup_k D_k^1)$ s.t. $(C \supset^{j-i-1}_{\downarrow} D_k{}')$ respectively.

Equivalent Relation	New Domain
(I) $X \supset Y$	
(1) $(*^m X \supset *^m Y) \leftrightarrow (X \supset Y)$	$*^m Y$
(2) $(*^m X \supset Y^{m+1}) \leftrightarrow (X \supset^{m}_{\downarrow} Y^{m+1})$	Y^{m+1}
(3) $(X^{m+1} \supset *^m Y)$:does'nt hold	----
(4) $(X^{m+1} \supset Y^{m+1})$:as it is	Y^{m+1}
(II) $X \cap Y \neq \emptyset$	
(1) $(*^m X \cap *^m Y \neq \emptyset) \leftrightarrow (X \cap Y \neq \emptyset)$	$*^m (X \cap Y)$
(2) $(*^m X \cap Y^{m+1} \neq \emptyset) \leftrightarrow (X \supset^{m-1}_{\downarrow} Y_k^m)$ for some k	$\bigcup_k Y_k^m$ s.t. $(X \supset^{m-1}_{\downarrow} Y_k^m)$
(3) $(X^{m+1} \cap *^m Y \neq \emptyset) \leftrightarrow (Y \supset^{m-1}_{\downarrow} X_k^m)$ for some k	$\bigcup_k X_k^m$ s.t. $(Y \supset^{m-1}_{\downarrow} X_k^m)$
(4) $(X^{m+1} \cap Y^{m+1} \neq \emptyset)$:as it is	$X^{m+1} \cap Y^{m+1}$

$$X^{m+1} = \{X_1^m, X_2^m, ---, X_M^m\}, \quad Y^{m+1} = \{Y_1^m, Y_2^m, ---, Y_N^m\}$$

Table 2 Equivalent set-theoretical relations and new domains

V Conclusion

A modification of predicate logic, called multi layered logic, has been discussed. It was designed as the knowledge representation language for describing the systems that uses engineering knowledge to support the hierarchical style of system analysis and design. A multi level data structure was used to represent the complex systems in which an abstract entity at a level of abstraction is formed as the collection of the entities at the next lower level. Multi layered logic adapts to this structure of the system and meets the conditions for supporting analysis and design of complex systems.

REFERENCES

[1] Chang, C. L. and R. C. T. Lee, "Symbolic Logic and Mechanical Theorem Proving" Academic Press, 1973.

[2] Enderton, H.B. "A mathematical Introduction to Logic." Academic Press, 1972.

[3] Hill, F. J., R. E. Swanson, M. Masud and Z. Navabi, "Structure Specification with a Procedural Hardware Description Language." IEEE Trans. Computers C-30:2(1981) 157-161.

[4] Lim, W. Y, P. "HISDL -- A Structure Description Language CACM 25:11 (1982) 823-830.

[5] Ohsuga, S. "A New Method of Model Description --- Use of Knowledge Base and Inference," In Proc. IFIP W.G.5.2. Working Conf. on 'CAD System Framework' June 1982. (To be published from North-Holland Pub. Co.,)

[6] Ohsuga, S. "Knowledge Based Man-Machine System," Proc. IFAC/IFIP/IFORS/IEA Conf. on Analysis, Design, and Evaluation of Man-Machine System Sept. 1982, pp.345-350.

DESCRIPTIONS AS CONSTRAINTS IN OBJECT-ORIENTED REPRESENTATION

Luc STEELS
Artificial Intelligence Laboratory
Vrije Universiteit Brussel
Pleinlaan 2 - 1050 Brussel (BELGIUM)

ABSTRACT

A motivation is given to introduce indefinite descriptions. Parts of a description language are presented. Mechanisms for the interpretation of indefinite descriptions are briefly discussed.

KEYWORDS : Knowledge representation, AI programming languages, object-oriented systems, description languages, reasoning, constrain! propagation.

1. INTRODUCTION

A form in LISP or a term in PROLOG can be viewed as a description, i.e. an expression which refers to an entity in a domain of discourse. Forms and terms are both definite descriptions. They are subject to two restrictions

(i) The uniqueness condition : A description may onJy denote a unique referent, it cannot be ambiguous. For exampJe, (Father John) is allowed because John can only have one father, but (Brother John) is not allowed because John may have more than one brother and it is therefore not clear which one is intended.

(ii) The computability condition: when the referent of a description is needed during interpretation, it should be computable. For example, the expression (Father John) must be computable when it is needed otherwise an error condition occurs.

Variables can also be viewed as descriptions. Typically, the uniqueness and computability conditions hold for then too : a variable may have only one value (within a given environment of course) and the value should be retrievable when needed, otherwise an error condition (unbound variable) results.

Descriptions that may have more than one possible referent or whose referent is not computable at the time it is needed are called indefinite. We want to develop a system where definite as well as indefinite descriptions can be used.

For example, (Greaterthan 10), which denotes an unknown number greater than 10, or (Divisor 12), which denotes a divisor of 12, i.e. 1,2,3,^,6 or 12, are indefinite descriptions which would be allowed in the system. Note that the conjuction of (Greaterthan 10) and (Divisor 12) uniquely refers to 12. In other words, the conjunction of indefinite descriptions may be definite.

A definite description provides a deterministic method to compute the referent when needed. An indefinite description expresses a constraint on its referent. Computation vith indefinite descriptions consists of operations over constraints. For example, (Greaterthan 10) expresses the constraint that the referent has to satisfy the predicate (lambda (X) (> 10 X)). (Divisor 12) introduces the constraint that the referent has to be a member of (1,2,3,)<,6,12). The predicate can be applied to filter members out of this set, so that tho conjunction of the two descriptions yields 12.

In recent years there have been some proposals that are relevant to this researdi Concepts like lazy evaluation (Henderson and Morris, 1976), or futures (Hewi11, 1977) make it possible to delay evaluation until needed or until sufficient information is available.

The concept of a logical variable as used in PROLOG (Kowalski, 1977) can be viewed as a way to relax the computability condition, because a logical variable can be used even though its referent is not known or only partially known. Note however that a logical variable can be bound to only one object at the time, which would make it necessary to backtrack 5 times before 12 yields a successful match in the previous example (but see Kornfeld, 1983).

Because the evaluation process needed to deal with indefinite descriptions resembles techniques used to implement constraint propagation, there are some interesting relations to constraint languages as well (cf. Borning (198O), Steele and Sussman (1980)

There are four motivations for studying indefinite descriptions. First they can be incorporated in ordinary programming languages, particularly in object-oriented languages in which definite descriptions already play an important role. Second they can be incorporated in knowledge representation languages, particularly in the recent generation of description languages (see e.g. Winograd (1982), Attardi and Simi (1982)).

Third, because indefinite descriptions feature prominently in natural language, the relation between formal languages and natural languages promises to become more transparent. Finally, indefinite descriptions can be used as a component of query languages for databases. It would make it possible that a user supplies constraints in the form of indefinite descriptions to refer to an object in the database-.

In this short paper a full treatment of the description system is impossible. Instead we only introduce some basic linguistic constructs and mention some properties of the evaluation mechanism.

1. DOMAIN VS. REFERENT

The first key idea is that we make a distinction between the domain of a description and its referent. The referent is the element denoted by the description. The domain is the set of possible elements out of which the referent has to be chosen. For the description 'uneven' the domain is the set of uneven numbers and the referent is an element out of this set - although it is unknown which one. The domain of 'the brother of John' is equal to all brothers of John. The referent is one element out of this set.

The distinction between domain and referent allows the construction of a clear set-theoretic semantics for a description language. By making it possible to give partial descriptions of the domain, it raises the expressive Power of the language Also the explicit representation of constraints on the domain makes the deduction more powerful. Often we know a lot about the domain of a description but not its referent. By operating at the level of domains, we can sometimes reduce the domain until it is a singleton, i.e. until there is a unique referent left. These points will be illustrated in the rest of the paper. First we introduce description types.

2. NAMES AND SPECIAL OBJECTS

The simplest form of a description is a name which uniquely indentifies an element in the domain of discourse. A name is an atom or a datastructure containing only names as elements. Sequences and sets are considered to he primitive datastructures. For example, JOHN, [JOHN MARY JAMES] and (JOHN MARY JAMES) are examples of names.

There are three special objects in the system. The undefined object or all-object, the null-thing or empty object, and the overdefined object. The names of these objects are T(top), NIL, and 1. (.bottom;. ?he domain of T is the universe of discourse. The domain of NIL is the empty set. ?he domain of 1 is overdefined. When at a :ertain point a subexpression refers to the)verdefined object, then the whole expres- ;ion refers to the overdefined object.

3. DESCRIPTIONS BASED ON CONCEPTS

The second type of descriptions is of the form <concept> or (<concept> < arg >. . . ;arg >) for n > 1. A concept may either be L function, in which case a unique referent :an be computed given referents as arguments. For example, (+5 10) is a description with referent 15. But a concept need tot be a function. For example, (Divisor 2) is a description with possible refe- •ents 1,2,3,4,6 and 12. Divisor is a concept but not a function.

4. DESCRIPTIVE CONNECTIVES

The descriptive connectives dAnd, dOr, Not, dEither and dAncinot are used to combine descriptions. They should not be contused with the propositional connectives used to combine statements in predicate calculus .

The descriptive connectives reflect set-theoretic relations between the domains of the component descriptions. The referent of (dAnd d1 d2) is an element out of the intersection of the domains of d1 and d2. The referent of(dOr d1 d2) is an element 5ut of the union of the domains of d1 and d2. The referent of (dEither dl d2) is an element of the domain of dl or of the domain of d2 but not of both. The referent of (dAndnot dl d2) is an element out of the set-theoretic difference between the domains of d1 and d2. (dNOT d) is an abbreviation of (dAndnot T d), i.e. dNot indicates a set-theoretic difference with the domain of the all-description. Thus (dNot female) is equivalent to (dAndnot T female).

It is easy to prove that T and NIL act as the identity and zero-element for these connectives. Analogues exists also for the other propositional laws, such as De Morgan's.

5. DESCRIPTIONS OF THE DOMAIN

There are a variety of things that could be know about the domain of a description. We want to have constructs that are able to express this partial information. Here are some examples :

ENUMERATION OF THE POSSIBLE MEMBERS.
An expression of the form (Element-of $X_1 .... X_n$ expresses the constraint that the referent has to be either X_1.... or X_n. For example, an alternative description for (Brother John) could be (Element-of George James).

PARTIAL ENUMERATION. A description of the form (INCLUDES X) expresses that the referent comes out of a domain that has X as a member. For example, if it is known that George is one of the brothers of John then (includes George) is an alternative description for (Brother John).

CARDINALITY OF DOMAIN. The description (Number-of X) with X an integer, indicates that the cardinality of the domain is equal to X. For example, if it is known that John has two brothers, then (Brother John) can be described as (Number-of 2).

The deduction rules for the descriptive connectives include rules for dealing with such domain descriptions. For example, the referent of (dAnd (Brother John) (Friend Frank)), where (Brother John) is (Element-of George James) and (Friend Frank) is (Element-of George Mary), is equal to George because George is the only element in the intersection of the domains of two descriptions.

6. VARIABLES

Variables are descriptions which start out as indefinite members of the universe and gradually assume their domain as constraints accumulate. Variables are preceded by the symbol : and are lexically scoped within the expression in which they occur, although they may occur anywhere in the description. For example, in (dAnd : Y 5), or its equivalent (dAnd 5 : Y), the referent of : Y will be equal to 5• Variables in the description system thus behave like logical variables.

7. CONVERSE DESCRIPTIONS

If (Father George) is a description for John, then (with Father John) is an alternative description for George. A description of the form (with <concept> <description> is called a converse description, because it denotes the converse of a concept.

8. EVALUATION

The goal of evaluation is to find the name of the referent of a description, i.e. a value. When a description is definite, its referent can be computed and evaluation proceeds as ordinary applicative evaluation When the description is indefinite, the result from evaluation is a collection of constraints on the referent, called a constraint cluster.

These constraints take the form of a predicate that the referent has to satisfy, generators which could start enumeration of the domain if needed, and constraints on the domain such as a list of its members a list of the elements not in the domain, a partial list of the members, the cardinality of the domain, etc.

The evaluator will attempt to proceed with the computation even though partial results are constraint clusters. Deduction rules for the descriptive connectives operate over constraint clusters.

For example, if the constraint clusters of two descriptions contains predicates, then the constraint-cluster of the conjunction of the two descriptions will contain the AND-conjunction of the two predicates. Thus (dAND (Greaterthan 10) (Lessthan 5)) results in ((Predicate (lambda (x)(and (> x 10)(< x 5)))))-

The evaluation process will also attempt to apply functions to constraint clusters. For example, when an explicit domain is known, computations can be performed by mapping the function. For example, (+(element-of 1 2)(element-of 3 4) is equal to (element-of 4 5 6).

Note however that it is possible to specify descriptions whose referent will not be computable because it would require the introduction of much more knowledge. For example, the description (dAnd Even Prime) has only one referent, namely 2, but this cannot be determined from knowing predicates or generators on the component descriptions themselves.

9. CONCLUSION

We argued for the introduction of indefinite descriptions, sketched some linguistic constructs and briefly indicated a possible evaluation.

REFERENCES

[1] Attardi and Simi (1982) Semantics of inheritance and attribution in the description system OMEGA. MIT-AI lab. Memo. 642

[2] Borning, A. (1979) THINGLAB, A. Constraint oriented simulation laboratory. Xerox Parc Report 55L~79-3

[3] Hewitt,Carl (1977) Viewing control structures as Patterns of Passing Messages. AI Journal 8, n° 3. PP. 323-364

[4] Henderson, P. and J. Morris (1976) A lazy evaluator. Proceedings of the 3d POPL Symposium, Atlanta George

[5] Kornfeld, B. (1983) Equality for PROLOG. IJCAI-83, Karslruhe

[6] Kowalski, R. (1978) Logic for problem Solving. North-Holland, Amsterdam

[7] Steele, G. and J. Sussman (1980) Constraints. MIT A.I. Las Memo 502

[8] Winograd, T. (1983) Language as a cognitive process. Prentice-Hall, Englewood Cliff.