COMMENTING PROOFS

James R, Geiser
Massachusetts Institute of Technology
Artificial Intelligence Laboratory
Cambridge, Massachusetts

Abstract

The goal of this investigation is the development of a semantics for 1 order theories based on certain new syntactic structures in formal proofs which derive from their pragmatic and semantic aspects. The present report is mainly concerned with these new syntactic structures, the motivation behind them, their technical definition and their basic properties. Their role in a new semantics for Intuitiom stc Peano Arithmetic is indicated in the last section.

1. Introduction

This paper constitutes a summary of a seminar entitled "Commenting Proofs" given at the Artificial Intelligence Laboratory at M, 1. T. during the spring of 1974. The seminar was concerned with new syntactical structures in formal proofs which derive from their pragmatic and semantic aspects. These structures can be used to provide a semantical interpretation for proofs quite distinct from that of Model Theory and rather more akin to the construction semantics for Intuitionistic Peano Arithmetic developed by K. Godel and the A. L procedural interpretation of the logical connectives and quantifiers a la C. Hewit's Planner Language. The work is actually a synthesis of elements from Yessenm-Volpin's foundational studies {l)and G. Sussman's work on building a "causal" history of a computation from the comments of the corresponding documented program (2).

For the most part we shall restrict ourselves to the context of Peano Arithmetic and the primative recuresive arithmetic of addition, multiplication and exponentiation in particular. At the end of this report a few remarks will be made on how these ideas are extended to a richer deductive environment.

In our work we distinguish between formal proofs (a sequence of sentences satisfying the usual syntactic criteria) and "real"proofs (a formal proof constructed by someone in accordance with a motivational system of goals). In the latter case each line may be commented by intrinsic (formal-syntactic) remarks and extrinsic (goal related) remarks. These remarks not only explain the purpose of a particular line, but at the same time establish connections with other lines (previous lines and lines yet to be constructed). In tact these remarks point to connections between the very signs that make up the lines of the proof. These connections, which, using Yessenin-Volpin's terminology, are called identificational connections or ids , are the links of a very detailed syntactic structure which resides implicitly in real proofs, a structure of causal chains connecting the occurences of symbols. This is the new syntactic structure mentioned in the first paragraph.

With this sort of information it becomes possible to answer a question like "what part of the term 11. *111* is responsible for the third stroke in the right hand side of the equation 11-111 = 111111 ?". Such a part will be called an ingredient of a term, consisting of a certain list structure whose atoms are the occurences of symbols in the term. If we think of a term t being evaluated to its numeral value ltl=ll...l then the ingredients represent the computational history of each stroke in ltl.

Much of the present report is concerned with defining the ingredients of a term and characterizing their dependence on the computational paths used to evaluate the term. The semantics which arises from this comes from considering a proof as encoding a procedure for manipulating list structures so that if, for example, the proof is of t=s then the encoded procedure induces an isomorphism from the ingredients of t onto the ingredients of s. While ingredients and the resulting semantics can be implimented it seems best at this point to treat these ideas theoretically and to be aware of their possible relevance to such concepts as consistency, relevant entailment, computational efficiency and methods of proof. We wont develope these connections in this report other than to briefly point out a connection between computational efficiency and the "normal" form of an ingredient in section 7.

j_-_Recursive Arithmetic.

Recursive Arithmetic consists syntactically of terms and equations, and deductively of computations (constucted by means of the recursion axioms and the substitution rule).

Terms are expressions formed from 1, + , " , exp, (,) by means of the following rules.

1) 1 is a term.

Z) If t is a term then so is tl.

3) If t and s are terms then so are (t + *a),* t • s, and exp(t, s) .

We shall use the symbols t, r, s, ..., t1,r1, s1,.....to denote terms. We shall make use of the usual conventions for ignoring paran - theses. Also note that our use of the word "term" doesn't allow for the occurences of free variables, i.e. they are always to be closed. *

Terms of the form 1, 11, ... $1^{(n)}$ (n concatenated strokes) are called numerals.

An equation is an expresion of the form t = s.

The Recursion Axiom Schemata:

Addition A1 $(t + 1) = t1$.
A2 $(t + s1) = (t + s)1$.
Multiplication M1 $t \cdot 1 = t$.
M2 $t \cdot (s1) = t \cdot s + t$.
Exponentiation E1 $exp(t, 1) = t$.
E2 $exp(t, s1) = exp(t, s) \cdot t$.

The Substitution Rule Schema:

Line LA $t = s(r)$
Line LB $r = r'$
Line LC $t = s(r')$.

Here s(r) denotes a term in which the term r has one or more indicated occurences, and s(r') denotes the term resulting from the replacement in s(r) of r by r' at the indicated occurences of r in s(r) . Note that this is more general than uniform substitution. We say that LC follows from LA and L.B by substitution.

A proof or computation is formally de- fined as a sequence of equations each of which is a instance of a recursion axiom or else follows by substitution from two previous equations. We consider two simple yet illuminating examples.

Example 2.1.
L1 $1 + 11 = (1 + 1)1$
L2 $1 + 1 = 11$
L3 $1 + 11 = 111$.

Example 2.2 .
L1 $1 \cdot 11 = 1 \cdot 1 + 1$
L2 $1 \cdot 1 = 1$
L3 $1 \cdot 11 = 1 + 1$
L4 $1 + 1 = 11$
L5 $1 \cdot 11 = 11$.

3. Commenting Proofs .

We begin commenting these computations by means of intrinsic and extrinsic remarks-- this terminology is after Sussman. Then we use these remarks to generate ids between the oc- curences of symbols in the computation. By tracing out chains of ids we can establish an ac- countability for every sign in the computation. This will at the same time make explicit the semantics of the computation, i.e. which oc- curences of symbols are synonomous and what are the computational roles of each sign.

The intrinsic or formal comments make note of :
1) if the line is an axiom; in which case a pointer is generated to the axiom schema in question;
2)if the line follows by substitution; in which case pointers are generated to the lines from which it follows and to the occurences of the term to be replaced.

The extrinsic or goal-related remarks consist of:
1) the top level goal statement (i. e. find the value of term t);
2) the assertion that line L (s =. s') is generated in order to simplify by substitution a term t(s) in a previous line L^1;
3) the assertion that line L," is the result of a substitution rule from lines L, L' whose pur- pose it was to achieve simplification by means of this substitution;
4) the assertion that the line matches the top level goal.

Example 2. 1 Commented.
The top level goal is to evaluate 1 + 11.
L1 $1 + 11 = (1 + 1)1$ **(Axiom A2) (Purpose is to simplify 1 + 11 of top level goal.)**
L2 $1 + 1 = 11$ **(Axiom A1) (Purpose is to simplify 1 + 1 of r. h. s. of L1 using substitution.)**
L3 $1 + 11 = 111$ **(Sub. L1, L2) (Purpose is to fulfill the goal of L2. L3 matches top level goal.)**

Example 2.2 is commented in a similar manner.

4. Identificational Connections.

We now add a third type of comment to the analyzed computation, namely we make note of the identificational connections (ids). First of all each axiom is to be accompanied by certain ids as follows.

A1 $t + 1 = t1$
A2 $t + s1 = (t + s)1$
M1 $t \cdot 1 = t$
M2 $t \cdot (s1) = t \cdot s + t$
E1 $exp(t, 1) = t$
E2 $exp(t, s1) = exp(t, s) \cdot t$.

For example, in A2 we would say that s and

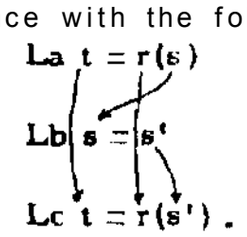
* The notational methods that we shall develope for +, * , and exp, will serve to handle all other primitive recursive functions.

t in the r. h. s. are rewritten from the s and t in the l.h. s. and this justifies their synonymity. On the other hand we also make an identificational connection between the two occurences of 1 and this constitutes our semantical interpretation of + in terms of the sucessor function xl.

Now consider M2. We say that both t's in the r.h. s. are rewritten from the l.h.s t. Similarly the s in the r.h. a. is rewritten from the s in the l.h. s, . However, we associate the second t in the r.h. s. with the stroke 1 of the l.h.s.——this is part of our semantical interpretation of * . Thus we are looking at m-n as saying add m to itself n times; in this computation n acts as a counter for the n different rewrittings of m. The strokes of m are the counter or control elements in this case.

Let A be an axiom and id(A) the set of ids associated with A. More explicitly: if a stroke p on the r.h. s is simply rewritten froma stroke q on the l.h. s. (without a control element) then put id(p,q) in id(A); if on the other hand p in the r. h, s. is rewritten! from q in the 1. h. s. under the control element q' and using an axiom for the operation f (either • or exp) then put id(p, (q, q', f)) in id(A) . *

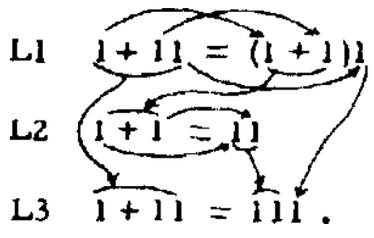
Ids come from the substitution rule in accordance with the following schema.



These ids actually come from the extrinsic comments associated with these lines: Lb's purpose is to simplify the indicated s in the r.h. s. of La. Hence the s in the], n. s. of Lb is rewritten from the s in the r. h. s. of La. Lc achieves the goal of Lb; hence the context r in the r.h. s, of Lc is rewritten from the context r in the r.h. s. of La and the s' in the r.h. s. of Lc is rewritten from the s* of the r. h. s, of Lb. Also, the 1. h. s, of Lc is rewritten from the l.h. s. of La.

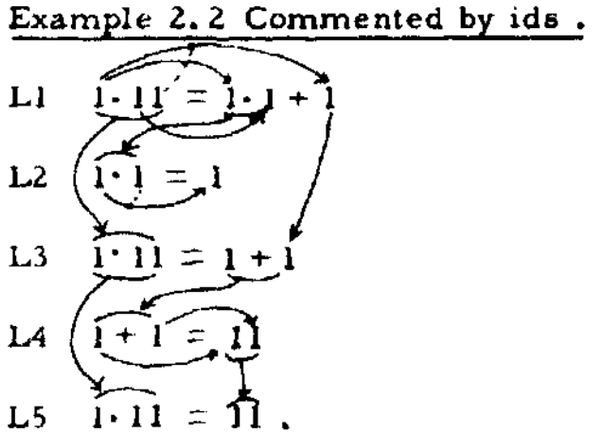
Fully commented by ids Examples 2.1 and *Z, Z* look like the following.

Example 2.1 Commented by ids.



* A set of ids like id(A) is usually taken symetrically, i. e. , if id(u, v) is in id(A) then so is id(v, u).

Example 2.2 Commented by ids .



We can now trace out paths of ids, thereby diagramming the computational relations between the different occurences of strokes in the proof. For example, in Example 2. 1, if we denote the occurences of strokes in L3 by p,p',p", q, q', q" (left to right) then we can see that p is connected to q, p' is connected to q' and p" is connected to q". Further more these are the only connections between the strokes of L3. This yields a very nice correspondence between the 1. h. s. and the r. h. s. strokes of L3.

The case of Example 2.2 is more complicated in that some of the occurences of strokes act as counter elements and when in this role they do not get rewritten. Lable the strokes in L5 p , p', p" q q' (left to right). Trace out a path from q or q' of L5 to a pattern of strokes p, p', and p" of L5, using the ids for the axioms and the substitution rule. We see that q traces out to the pattern (p, p', -) and q' traces out to the pattern (p, p", *). Thus we are lead to say that q is rewritten from p under the control element p' while q' is rewritten from p under the control element p". (p, p', •) and (p, p", •) are the patterns in 1.11 of L5 "responsible" for q and q' respectively in this computation. These patterns, which will be precisely defined in section 6, we call the ingredients of 1.11. Generally speaking, when a term t is evaluated (i.e. proved equal to some numeral n) we may identify the ingredients of t responsible for each stroke in n . In section 7 we determine how the ingredients depend on the computation path used to evaluate the term t.

5. Computation Paths .

On the previous pages we have presented two examples in some detail inorder to give an intuitive picture of what is happening with proofs, comments and ids. We shall now turn to a detailed study of the possible computation paths from a term.

We shall restrict our attention to computations which have the following (standard) form:

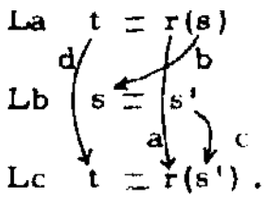
$$L1 \quad t_1 = t_2$$

L2i-1 $t_1 = t_{i+1}$
L2i $s_i = s_i'$
L2i+1 $t_1 = t_{i+2}$

L2n-1 $t_1 = t_n$.

where L1 is an axiom and for i = 1, . . . , n-2, L2i+1 follows from LZi-1 and L2i by substitution. We shall assume that these computations have been commented and the appropriate ids have been made.

The sequence of terms t (=t_1), t_2, . . . , t_n is called a computation path from t to t_n. Consider a step of the computation (called a simple reduction)



We write r(s) - r (s') to indicate that r (s') has been obtained from r(s) by a simple reduction. If u = v is an axiom we shall also write u -• v and call it a simple reduction.

Define ID(r(s)—>r(s')) to be the set of derived ids between the signs of r(s) and r(s'). Specifically :

1) If p is an occurence of a stroke in r(s') which is rewritten from q in r(s) via the id a then id(p.q) is in ID(r(s)—*r(s'))-

2) If p is an occurence of a stroke in r(s') which is rewritten from q in s' of Lb via c and q is rewritten from q' in s of the 1. h. s. of Lb (i. e. the id id(q',q) is in id(s = s'), see section 4), and q' is rewritten from q" in s in the r. h. s. of La via b, then id(p,q^n) is in ID(r(s)—>r(s')).

3) If p is an occurence of a stroke in r(s') and is rewritten from q in s' of Lb via c and q is rewritten from q* in s of the 1. h. s. of Lb under the control of q" in the l.h. s. of Lb for the function f (i. e. id((q', q", f), q) is in id(s — *s')),* and q'and q" are rewritten from q* and q** respectively in s in the r. h. s. of La via b then id(p, (q*, q**, f)) is in ID(r(s)—>r(s')l

4) If u — v is an axiom define ID(u — *-v) to be equal to id(u - v).

Every occurence of a stroke p in r(s') is associated through ID(r(s) —*r(s')) with a unique stroke q in r(sj of La or a unique pattern ($q\#_r q^{**}$, f) of strokes in r(s).

If tj, . . . t_n is a computation path P and t_n is a numeral then P is called an evaluation path and t_n is called the value of ti w. r. t. P .

Define T(t) to be the set of all terms occuring in the computation paths from t; the relation s—>r determines a partial ordering of T(t) which we will now investigate.

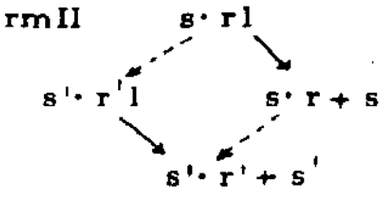
The notation s-->r denotes a computation path frim s to r. We say that t_1-->t_n is a subterm path if none of the reductions t_i : - t_{i-1} involve the main function (outer most function symbol in polish notation) .

Definition5. 1 A loop consists of two different computation paths going from a term t to a term t'. A diamond is a loop which has either form 1 or II below. We illustrate this using • as the main function symbol.

Form I

(where the indicated paths are subterm computation paths)

FormII

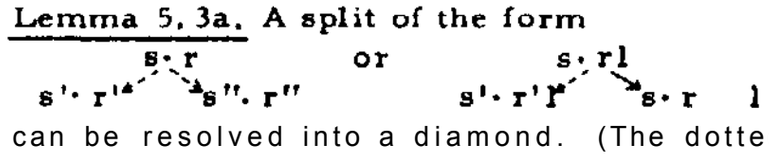


(where both broken pathe are subterm paths; we may assume that each involves that same associated subterm paths s-->s' and r-->r' .

Definition 5. 2 Two computation paths are simple variants if they differ by a diamond, i.e. they look like

Two paths are homologous if there is a sequence of computation paths PI, P2, ... Pn such that PI and Pn are the two paths in question and for i - 1, . . . , n- 1, Pi and Pi- 1 are simple variants. Note that homologous evaluation paths assign the same value to a term.

Lemma 5, 3a. A split of the form

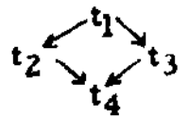


can be resolved into a diamond. (The dotted paths are subterm paths. Multiplication is serving as a paradigm case.)

Lemma 5. 3b, Any split can be resolved into a loop.

Lemma 5. 3 is an anologue to the Church-Rosser theorem for the Lamda Calculus. The proof of this and other results is by an induction argument on the rank rk(t) of a term, where rk is an integer valued function (primitive recursive) defined so that a) if s is a subterm of t then rk(s) < jjc(t) and b) if s—♦ t then j-kjt) < Hk(s). The existence of such a function shows us that any computation path from t has less than or equal to rk(t) steps. If we were using all primitive recursive functions then such a rank function could be

general recursive but not primitive recursive, e.g. like Ackerman's function. The organization of the proof is to dovetail 5. 3a and 5. 3b, first proving 5. 3a for rk(t) < n and then 5. 3b for rk(t) < n. The proof is somewhat more complicated than the usual one because we do not require substitution to be uniform, a condition which would considerably simplify the structure of T(t), namely any split of a major connective would always immediately resolve:



6. Ingredients.

Definition 6. 1. Let q1, . . . , q denote the occurences of strokes in the term t. Ing(t) denotes the set of list expressions obtained from q1, . . . q_n, " , and exp, according to the following rules.

1) 9 ,.....q_n are in Ing(t).
2) If i_ and j_ are in Ing(t) and f is • or exp then

The members of Ing(t) are called the abstract ingredients oft, and q,,...,q are called the simple ingredients of t. Ing (t) denotes the set of simple ingredients of t. In the expression (i,j,f), j *is called the* control element and i is called the raw material. To understand the motivation behind this definition let us once again *consider* the recursive definition of u. v : u J = u and u-vl r u. v + u. Thus v act as a control element and u as a fixed parameter. Correspondingly, if i and j were ingredients in r and s resp. then in the term t equal r* s, (i,j,f) would be an ingredient in t with j as the control element and i as the "raw material" to be recopied.

In the analysis of ingredients and how they change as we trace out the ids in a computation it soon becomes clear that these changes tend to respect the internal structure of ingredients and can be characterized as "homomorphisms" of this structure. So we make a definition.

Definition 6. Z. A mapping H: Ing(t)—*Ing(s) is a homomorphism (horn) iff for all (i, j, f) in Ing(t) H((ifji,0> = <H(iKH£),f) .

Facts about Homomorphisms.
6. 3a If H,G: Ing(t)-*Ing(s) are homomorphisms which agree on $Ing_Q(t)$ then H = G.
6. 3b Any map H : $Ing_0(t)$-—*Ing(6) extends to a unique horn from Ing(t) into Ing(s).
6. 3c Horn H is 1-1 on Ing(t) iff H is 1-1 on $Ing_0(t)$.

There are infinitely many abstract ingredients in Ing(t) only some of which represent real computations. We now use the idea of homomorphism tosingle these out in a precise manner.

We associate with a simple reduction t→s the following hom H[t→s]. Let ID(t→s) be the set of ids accompanying t→s as defined in section 5. H[t→s] is the unique hom from Ing(s) into Ing(t) determined by the following (here q denotes any simple ingredient in $Ing_o(s)$).

$$H[t\to s](q) = \begin{cases} p & \text{if } id(p,q) \text{ is in } ID(t\to s), \\ (p,p',f) & \text{if } id((p,p',f),q) \text{ is in } ID(t\to s). \end{cases}$$

H[t→s] is 1-1 on $Ing_o(s)$ and hence is 1-1 on all of Ing(s). We can now define the computationally meaningful ingredients.

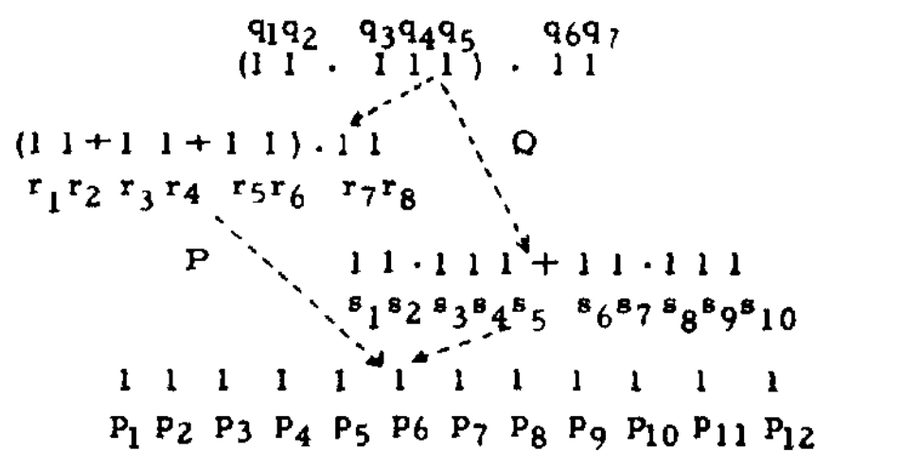
Definition 6. 4. Let P = $t_1,\ldots,t_n$ be an evaluation path for t; thus |t| = t_n. Define the hom $H[P] = H[t_1\to t_2]\circ H[t_2\to t_3]\circ\ldots\circ H[t_{n-1}\to t_n]$. The set of (real) ingredients of t w. r. t. P is the set $H[P](Ing_o(t_n))$ and is denoted by Ing(t;P).

Note that H[P] is a 1-1 map of $Ing_o(|t|)$ into Ing(t). So the cardinality of Ing(t;P) is equal to the integer denoted by the numeral |t|. We think of H[P](q) as the unique "computational pattern" in t which is "responsible" for q via P.

7. Invariance of Ingredients.

How does H[P] depend on P? This is answered in Theorem 7. 6, but first consider the following example.

Example 7.1.



Computing H[P](p9) and H[Q](p9) we get

H[P](p9) = i = ((q1, q4, •), q7, •) and

H[Q](p9) = j = ((q1, q7, •), (q4, q7, •), •).

The basic difference bewtween i and j is the temporal order in which the control elements q4 and q7 are operating and not the ultimate control relationships.

The above relationship between i and j is made into a basic equivalence relation.

Definition 7. 2. Let i and j be ingredients in

Ing(t). We define $\underline{i} \approx \underline{j}$ iff there is a sequence of pairs $(\underline{i}_1, \underline{j}_1), \ldots, (\underline{i}_n, \underline{j}_n)$ of ingredients of Ing(t) such that $\underline{i} = \underline{i}_n$, $\underline{j} = \underline{j}_n$, and for $k = 1, \ldots, n$ either

1) $\underline{i}_k = \underline{j}_k$,

2) there exists ingredients $\underline{u}$, $\underline{v}$, $\underline{w}$ and function symbols f, g such that $\{\underline{i}_k, \underline{j}_k\}$ equals $\{((\underline{u}, \underline{v}, f), \underline{w}, g), ((\underline{u}, \underline{w}, g), (\underline{v}, \underline{w}, g), f)\}$, or

3) there exists k', k'' less than k such that $\underline{i}_k = (\underline{i}_{k'}, \underline{i}_{k''}, f)$ and $\underline{j}_k = (\underline{j}_{k'}, \underline{j}_{k''}, f)$.

Our Example 7. 1 suggests that as we move from one computation path P to a simple variant P' of P we will find that HCPj(q) is changed into an equivalent ingredient H[P'3(q); this is the content of Theorem 7. 7 .

7. 3 Facts about ≈ .

7, 3a) ≈ is an equivalence relation on Ing(t).

7, 3b) If $\underline{i} \approx \underline{j}$ and $\underline{i}' \approx \underline{j}'$ then $(\underline{i}, \underline{j}, f)$ is equivalent to $(\underline{i}', \underline{j}', f)$.

7, 3c) If $\underline{i}_k \approx \underline{j}_k$ for $k = 1, 2, 3$ then $((\underline{i}_1, \underline{i}_2, f), \underline{i}_3, g) \approx ((\underline{j}_1, \underline{j}_3, g), (\underline{j}_2, \underline{j}_3, g), f)$.

Theorem 7. 4. Let H : Ing(s) ⟶ Ing(t) be a hom. Then H preserves ≈ , i. e. if $\underline{i}$ and $\underline{j}$ are in Ing(s) and $\underline{i} \approx \underline{j}$ then $H(\underline{i}) \approx H(\underline{j})$.

Theorem 7. 5. Let G, H : Ing(s) ⟶ Ing(t) be two homs and suppose that for all p in $\text{Ing}_o(s)$, G(p) ≈ H(p). Then $G(\underline{i}) \approx H(\underline{i})$ for all $\underline{i}$ in $\text{Ing}_o(s)$.

Lemma 7. 6. Let t be any term with rk(t) ≤ n.

I. If P and Q are computation paths from t to t' forming a diamond and $\underline{i}$ is a real ingredient in Ing(t') then $H[P](\underline{i}) \approx H[Q](\underline{i})$.

II. If P and Q are computation paths from t to t' and $\underline{i}$ is a real ingredient of Ing(t') then $H[P](\underline{i}) \approx H[Q](\underline{i})$.

The Lemma *is* proved by induction on n and dovetailing I and II. Lemma 5. 3 plays a major role. As an immediate consequence of this Lemma we get the main result of this section.

Theorem 7. 7. (Invariance of Ingredients.) Let P and Q be two evaluation paths for t. Then for all p in $\text{Ing}_o(t)$, H[P]{p) = H[Q]{p) .

There is a natural direction of simplification built into the equivalence relation = , namely we will say that ((i ,k, f), (j, k, f), g) reduces to ((i ,j g), k, f) . We then define an ingredient to be in normal if it can't be reduced nor does it contain any ingredients which can be so reduced. If one tries to make a cost estimate for various computation paths it appears that if H[P]{q) is in normal form then P is the least expensive computation path. Such estimates are not so easy to make in a reliable way but the examples which have been studied are quite suggestive.

8. Final Remarks.

The ideas presented in this paper can straightfowardly be applied to other deductive-computational systems such as the Lamda Calculus and Curry's Combinatorial Logic.

The next step in this investigation extends our analysis to the full 1st Order Intui*tionistic Theory of* Peano Arithmetic. The semantics developed provides a means whereby with each theorem A we can associate a collection M(A) of procedures extracted from the proofs of A. This will be the subject of a second paper being prepared under an NSF grant through Boston University.

Two examples will give some idea of how this semantics works.

Suppose A is the sentence $t = |t|$. Then M(A) will contain the procedures which compute the isomorphisms $H[P] : \text{Ing}_o(|t|) \longrightarrow \text{Ing}(t)$ determined by each evaluation path P for t.

Suppose A is the sentence $B \supset C$. Then a procedure in M(A) will be a method for convertint a procedure in M(B) into a procedure in M(B). If A is the sentence $B \vee C$ then a procedure in M(A) would construct a sentence D (which is either B or C) and a procedure in M(D).

Negation is defined in terms of implication $\sim A =: A \supset 1 = 11$. This at first seems a little odd because it says that you have a procedure which will transform any procedure in M(A) into a procedure in M(1 = 11), and, ofcourse, with the means that we allow there are no procedures in M(1 = 11). If we look at a concrete example like $11 = 111 \supset 1 = 11$, this becomes less mysterious since with a little thought we can see how a set of rules, proportedly establishing an isomorphism between $\text{Ing}_o(11)$ and $\text{Ing}_o(111)$ could be actually modified so as to establish an isomorphism between $\text{Ing}_o(1)$ and $\text{Ing}_o(11)$.

The Induction Schema works out very nicely. For if we have procedures M, M' in M(A(1)) and M($\forall x (A(x) \supset A(x1))$) respectively, then our procedure in M($\forall x A(x)$) starts with M and makes a recursive call to M'.

References.

(1) Yessenin-Volpin, A.S., "UUraintuition* ism and the Antitraditional Program for the Foundations of Mathematics*', Procedings of the Summer conference on Intuitionism and Proof Theory at Buffalo, New York, 1968, North Holland.

Z) Sussman, G., "A Computational Model of Skill Aquisition", Doctoral Thesis, MIT, 1973.

D3FIKITI0N THEORY AS T^A5I5 FOR A CRMTIV^ PROBLEM SOLVER

H.Andreka, T.Gergely, I.Nemeti
ftungarian Academy of sciences
Budapest, ungary

Abstract

In this paper the application of some deep theorems of mathematical logic is shown in the field of artificial intelligence. Namely, using some of the results of definition theory we give the mathematical base to systems for automatic designing. /SAD/, These systems are capable of solving constructive tasks of such kind that need some creativity from the psychological point of view. Above tasks contain the imtlicite description of the object to be contructed. First of all that unit is investigated at SAD which provides an explicit definition to the circumscribed object.

Introduction

One of the main directions in research of artificial intelligence is developing problem solving systems namely, systems for automatic designing /SAD/, Their practical importance is invaluable. These systems are capable to solve constructive tasks, A task is constructive if the unknown is some kind of an object of which characteristics are described in the conditions of the ta3k. Two kinds of these are distinguished:

1. The objects to be constructed are defined explicitly:
   a/ well-defined task
   b/ incompletely defined task - here the conditions provide an incomplete description of the object
2. The objects to be constructed are defined implicitly.
   In these take the objects are not named only certain expectations are given about them.

Designing tasks appearing on the expectations of a non-professional customer belong to latter type. It can't be expected from him to give an explicit definition of a required program e.g. with the input-output relation. All he can do is to give some hints on his own expectations towards some "programlike" thing.
Similar problems occur at decision making where information is implicitly connected with the question to be decided about.

A SAD capable of solving the 3econd type constructive task, must consist of the following two basic oomponents:

1. High-level problem defining unit which provides an explicit definition to the implicit object description
2. Solving unit which carries out the explicitly defined task

Mathematical logic and its model theory provides plenty of facilities in SAD research. In our present study we introduce the usefulness of definition theory, an intensively developing filed of model-theory, from the point of view of SAD.

Basic definitions

The following triple form a language: (syntax, the set of possible worlds, validity), or formally L =(*F,* M, F).

A type t is a pair of functions, i.e. $t=\langle t', t''\rangle$ such that

1. $Rgt' \subseteq \omega \setminus \{0\}$ where $\omega = \{0,1,2,\ldots\}$
2. $Rgt'' \subseteq \omega$
3. $Dot' \cap Dot'' = \emptyset$ where $\emptyset$ denotes the empty set.

Mere Dot and Rgt are the domain and the range of t respectively. Dot' is the set of relation symbols and Dot" is the set of function symbols.

In the followings we suppose that a t-type first-order langugage $L^t=\langle F^t, M^t, \models\rangle$

is given. Here M^t is the set of t-type structures. A t-type structure $\mathcal{A}$ is a function for which

1. $\mathcal{A}(0) \stackrel{df}{=} A$ is the universe of the structure
2. $\mathcal{A}(R) \subseteq {}^{t'(R)}A$ for each relation symbol $R \in Dot^1$
3. $\mathcal{A}(F): {}^{t''(F)}A \to A$ for each function symbol $F \in Dot^2$ and if $t''(F) = 0$ then $\mathcal{A}(F) \in A$

Aboves are to be found in more details in [1] Notations of common knowledge are also to be found there.

Prom now on when program is being discussed relation symbols will be used in describing the camputer programs where such symbols may show what relation the input-output should have. This descriptive method provides a far more natural handling of the programs than the descriptions of programs by functions, since this approach is more close to the intuition of the non-programer customers.

Intuitive description of SA*D* based on the definition theory

Let $\Gamma \subseteq {}_1F^t$ be a set of first-order formulas which provides the knowledge of a discripline within that designing will occur. S.E.P *P* provides the semantics of a programing language and the properties of different implemented programs.

The customer give3 hi3 requests with the help of a set of formulas Σ . This implicitly defines one or more relation symbols and/or function symbols which do not occur in *Dot' U* Dot" . in the followings without limiting generality, we supposec that Σ gives the implicit definition of only one relation symbol "P. E.g. Σ gives the implicit definition of such a program of which input and output are in relation P. Let $\Sigma'(P)$ denote the set of formulas defining the relation P implicitly.

L $t_P \stackrel{df}{=} \langle t'u(P,n),t''\rangle$ bee extension o f the type t and ${}_1F^{t_P}$ be the syntax of the first-order language extended by relation P. Thus $\Sigma'(P) \subseteq {}_1F^{t_P}$.

T carry out the design of the required object we have to give its explicit description by a formula of ${}_1F$. bo as to have the required program written in our programing language we have to find such a formula from F which defines *V* explicitly.

Let $P, P' \notin Dot' \cup Dot''$ be two n-placed relation symbols and $\Sigma'(P') \subseteq {}_1F^{t_{P'}}$ . .Ve say that $\Sigma'(P)$ defines P <u>implicitly</u> if
$\Sigma'(P) \cup \Sigma'(P') \models \forall \bar{x}^{(n)} (P(\bar{x}^{(n)}) \to P'(\bar{x}^{(n)}))$
where $\bar{x}^{(n)} \stackrel{df}{=} (x_1, \ldots, x_n)$ and $\forall \bar{x}^{(n)} \stackrel{df}{=} \forall x_1 \ldots \forall x_n$.
.Ve note that $\Sigma'(P')$ is obtained from by replacing P everywhere by P'.

.Ve give an equivalent definition to this:
Let $(\mathcal{A},R) \stackrel{df}{=} \mathcal{A} \cup \{\langle P,R\rangle\}$ where $\mathcal{A} \in M^t$, $R \subseteq {}^nA$
Given any models $(\mathcal{A},R)$ and $(\mathcal{A},R')$ for $\Sigma'(P)$ then $R = R'$. This means that $\Sigma'(P)$ implicitly defines P if for any model $\mathcal{A} \in M^t$ there is at most one n-placed relation R interpreting the relation symbol P such that $(\mathcal{A},R) \models \Sigma'(P)$.

Let $\varphi \in {}_1F^t$. If it has n free variables then we use the notation $\varphi[\bar{x}^{(n)}]$.

$\Sigma'(P)$ explicitly defines the relation P if there is a formula $\varphi[\bar{x}^{(n)}] \in {}_1F^t$ for which
$\Sigma'(P) \models \forall \bar{x}^{(n)} (P(\bar{x}^{(n)}) \leftrightarrow \varphi[\bar{x}^{(n)}])$.
Replacing P by φ in the set of formulas Σ' everywhere we obtain $\Sigma'(\varphi)$. For $\Sigma'(\varphi)$ the following is true:
(*) $\Sigma'(\varphi) \equiv \Sigma'(P) \cup \{\forall \bar{x}^{(n)} (P(\bar{x}^{(n)}) \leftrightarrow \varphi[\bar{x}^{(n)}])\}$

where $\equiv$ is the symbol of semantical equivalence.

what is the task of a high-level problem defining unit supposed to be at SAD? It has to find a definition $\theta \in {}_1F^t$ on the base of Γ knowledge to the requested expecta-

tion of the customer given by $\Sigma'(P)$ so that

$$\Gamma \cup \{\forall \bar{x}^{(n)}(P(\bar{x}^{(n)}) \leftrightarrow \theta[\bar{x}^{(n)}])\} \vdash \Sigma'(P) \quad .$$

In other words using $(*)$ such formula $\theta[\bar{x}^{(n)}] \in F^*$ has to be found for which $\Gamma \vdash \Sigma'(\theta)$.

This task results in the following questions:

1. Does a formula θ exixt to Γ so $\Gamma \vdash \Sigma'(\theta)$. If such doesn't exist then could Γ be extended, let's say, to a Γ' $(\Gamma' \subseteq F^*)$ so as to have the required formula θ existing such that

$$\Gamma' \vdash \Sigma'(\theta) \quad .$$

This procedure can be done with the help of a system consisting of a theorem prover and of an inductive hypothesis generator. First it will examine the truth of $\Gamma \cup \{\neg \wedge \Sigma\} \vdash \varphi \wedge \neg\varphi$ /here $\wedge\Sigma$ is obtained so that all the formulas of Σ are linked with the "and" connective $\wedge$ /. If this isn't true then we examine whether $\Gamma' \cup \{\neg \wedge \Sigma\} \vdash \varphi \wedge \neg\varphi$ is true. If this isn't so then we take another extension Γ'' etc.

We note that selecting Γ', Γ'' suppose an oriented inductivity.

The following problem belongs to here also. Is it true that all certain characteristic model $\mathcal{A}$ $(\mathcal{A} \in M^{\Gamma})$ of a set of formulas Γ becomes a model of $\Sigma(\theta)$ too, i.e. is it true that

$$\mathcal{A} \vdash \Sigma'(\theta) \quad .$$

Let us suppose that the existense of $\theta \in F^*$ is proved or that taking the risk of a possible negative answer we suppose the existence of θ . In this case the following question appears.

2. How can we obtain the suitable formula θ from set of formulas $\Gamma \cup \Sigma(P)$? Here we show some of the possible ways of producing formula θ .

a/ $\Gamma \cup \Sigma(P) \vdash \forall \bar{x}^{(n)}(P(\bar{x}^{(n)}) \leftrightarrow \theta[\bar{x}^{(n)}])$

b/ $\Gamma \cup \Sigma(P) \vdash \exists \bar{\sigma}^{(m)} \forall \bar{x}^{(n)}(P(\bar{x}^{(n)}) \leftrightarrow \theta[\bar{x}^{(n)} \bar{\sigma}^{(m)}])$

i.e. the definition of P is parametrically given by the set of formulas $\Sigma'(P)$. Here $\bar{\sigma}^{(m)} = (\sigma_1, \ldots, \sigma_m)$, $\exists \bar{\sigma}^{(m)} = \exists \sigma_1 \ldots \exists \sigma_m$.

c/ $\Gamma \cup \Sigma(P) \models \bigvee_{1 \le i \le k} \forall \bar{x}^{(n)}(P(\bar{x}^{(n)}) \leftrightarrow \theta_i[\bar{x}^{(n)}])$

i.e. $\Sigma'(P)$ defines P explicitly up to disjunction.

d/ $\Gamma \cup \Sigma'(P) \vdash \bigvee_{1 \le i \le k} \exists \bar{\sigma}^{(m)} \forall \bar{x}^{(n)}(P(\bar{x}^{(n)}) \leftrightarrow \theta_i[\bar{x}^{(n)} \bar{\sigma}^{(m)}])$

i.e. $\Sigma'(P)$ defines P explicitly up to parameters and disjunction.

It might happen that the set of formulas Γ has to be extended till Γ' as it is mentioned in **1.** so as to define θ .

In that case if set of formulas $\Sigma'(P)$ is too weak then, similarly to the methods described in [2] we have to find such a formula θ for which

$$\Gamma \cup \{\forall \bar{x}^{(n)}(P(\bar{x}^{(n)}) \leftrightarrow \theta[\bar{x}^{(n)}])\} \vdash \Sigma' \quad .$$

The set of formulas Γ can be extended here too if found necessary.

In that case if answer to question 1. is positive the following statement is true.

Lemma: a/ if $\Gamma \cup \Sigma(P) \vdash \forall \bar{x}^{(n)}(P(\bar{x}^{(n)}) \leftrightarrow \theta[\bar{x}^{(n)}])$ then $\Gamma \vdash \Sigma'(\theta)$

b/ if $\Gamma \cup \Sigma(P) \vdash \bigvee_{1 \le i \le k} \forall \bar{x}^{(n)}(P(\bar{x}^{(n)}) \leftrightarrow \theta_i[\bar{x}^{(n)}])$

then $\Gamma \vdash \Sigma'(\theta_1)$ or $\Gamma \vdash \Sigma'(\theta_2)$, ..., or $\Gamma \vdash \Sigma'(\theta_k)$.

We note here that we have to try the θ_i $(1 \le i \le k)$ in b/ till the first formula where the statement stands for true.

If the answer to question 1. is negative then the knowledge within the disciplines defined by Γ is not enough for the explicit description of the required object.

On the basis of aboves a "high-level"

problem defining unit of SAD should operate the following way.

The basic knowledge of SAD is provided by set of formulas Γ • The customer gives his required object description by the help of set of formulas $\Sigma(P)$
As a first step the unit has to find an exact answer for the existence of θ , but since it ic to complicated a task the following way is chosen. Firsb the system controls whether $\Sigma(P)$ contradicts to knowledge Γ , i.e. it tries to deduce the identically false $(\varphi \wedge \neg\varphi)$ from $\Gamma \cup \Sigma$. If this doesn't suceed within a present time period then the system presupposes the existence of a formula and it will proceed onto the 2. task, i.e. producing θ
Let us suppose that we succeeded in producing such a formula. It is followed by tis trying:

$$\Gamma \vdash \Sigma(\theta)$$

If this is true then θ really becomes the requirements of the customer if not, then it may be supposed that the knowledge Γ of the SAD is not satisfactory for defining θ . Therefore Γ has to be extended till Γ' and aboves have to be repeated nov; for set of formulas Γ'
The system will go on with this either until it proves the impossibility of $\Sigma(P)$ on the basis of the extended set of formulas or, it succeeds to produce formula θ • Of course the system goes on with trying only for a fixed time.
W'enote that the extension of set of formulas Γ need inductive logical means from the system.

Now we shall see that case when $\Sigma(P)$ defines *V* only up to the disjunction, tha; is when

$$\Gamma \cup \Sigma(P) \models \bigvee_{1 \le i \le k} \forall \bar{x}^{(n)} (P(\bar{x}^{(n)}) \leftrightarrow \theta_i[\bar{x}^{(n)}])$$

The so obtaimed formulas $\theta_i[\bar{x}^{(n)}]$ $(1 \le i \le k)$ have to be controlled one by one. So $\Gamma \vdash \Sigma(\theta_1)$ or $\Gamma \vdash \Sigma(\theta_2)$,..., or $\Gamma \vdash \Sigma(\theta_k)$.
This control goes on until the first θ_i for which $\Gamma \vdash \Sigma(\theta_i)$, If neither θ_i satisfies above condition then it might be supposed that the knowledge Γ is not satisfactory. In this case the procedure goe3 on similarly, i.e. Γ is extended until Γ', etc.

Useful theorems of definition theory

In this chapter we introduce those theorems of definition theory without proof which provide the explicit definition of P on the basis of $\Sigma(P)$ and Γ • Their proofs can be found in [1]. It is expected to obtain different types of theorems depending on the strenght of $\Sigma(P)$. Vle begin with the theory containing the weakest conditions for $\Sigma(P)$.

If $\Sigma(P)$, Γ and a model $\mathfrak{A}$ is given then the conditions of the theorems contain either that how many relations $R \subseteq {}^n A$ are there for which $(\mathfrak{A}, R) \models \Sigma(P)$; or that how many such relations $R' \subseteq {}^n A$ are there to such a relation $R \subseteq {}^n A$ so as

$$(\mathfrak{A}, R') \cong (\mathfrak{A}, R)$$

1.Theorem /Chang - Makkai Theorem/. If for every model $(\mathfrak{A}, R)$ /for which $|A| > \omega$/ of $\Sigma \cup \Gamma$:

$$|\{R' : (\mathfrak{A}, R) \cong (\mathfrak{A}, R')\}| < 2^{|A|}$$

then there are a finite number of parametric formulas $\theta_1[\bar{x}^{(n)} \bar{\sigma}^{(m)}]$, $\theta_2[\bar{x}^{(n)}, \bar{\sigma}^{m}]$,..., $\theta_k[\bar{x}^{(n)}, \bar{\sigma}^{(m)}]$ of $,F^t$ such that

$$\Gamma \cup \Sigma(P) \models \bigvee_{1 \le i \le k} \exists \bar{\sigma}^{(m)} \forall \bar{x}^{(n)} (P(\bar{x}^{(n)}) \leftrightarrow \theta_i[\bar{x}^{(n)} \bar{\sigma}^{(m)}])$$

The theorem intuitively states if $\Sigma(P)$ circumscribes the relation P in some measure then there exists a parameter- - v e c $(\sigma_1, \ldots, \sigma_m)$ and there are formulas $\theta_i[\bar{x}^{(n)} \bar{\sigma}^{(m)}]$ $(1 \le i \le k)$ of $,F^t$ such that one of them gives the definition of P. In other words the set of formulas $\Sigma(P)$ defines P explicitly up to parameters and disjunction.

Theorem 2. If set of formulas $\Sigma'(P)$ is such that to each model $\mathcal{A} \in M^t$ it is

$$|\{R \ (\mathcal{A},R) \models \Sigma\}| < 2^{|A|}$$

then there exists a finite number of first-order parametric formulas Θ_i $(1 \le i \le k)$ so that

$$\Gamma \cup \Sigma(P) \models \bigvee_{1 \le i \le k} \exists \bar{v}^{(m)} \forall \bar{x}^{(n)} (P(\bar{x}^{(n)}) \leftrightarrow \Theta_i[\bar{x}^{(n)} \bar{v}^{(m)}])$$

The intuitive meaning of the theorem is as it follows: if the number of relations satisfying set of formulas $\Sigma'(P)$ is less than the number of all possible relations then up to disjunction $\Sigma'(P)$ parametrically defines relation P. The condition of the theorems claims that not all the possible relations should carry the characteristics described by the set of formulas $\Sigma'(P)$.

Above theorems /Theorems 1. and 2./ are true also for that case when the number of the suitable relations is less than not $2^{|A|}$, but $|A|^+$. e. in this case there exists a finite number of first--order parametric formula and such a parametervector that one of the formulas will give the definition of relation P by the suitable parametervector.

Now let us see those cases when the possible number of relations satisfying $\Sigma'(P)$ are finite in the models.

Theorem 3. If for every model $(\mathcal{A},R)$ $(\mathcal{A} \in M^t)$ of $\Sigma'(P) \cup \Sigma$ it is true that

$$|\{R' : (\mathcal{A},R) \cong (\mathcal{A},R')\}| < \omega$$

then there exists such a $k < \omega$ and there are such formulas $\sigma[\bar{v}^{(m)}]$, $\Theta_i[\bar{x}^{(n)} \bar{v}^{(m)}]$ $(1 \le i \le k)$ in F^t that

$$\Gamma \cup \Sigma'(P) \models \exists \bar{v}^{(m)} (\sigma[\bar{v}^{(m)}]) \wedge$$

$$\wedge \forall \bar{v}^{(m)} [\sigma[\bar{v}^{(m)}] \rightarrow$$

$$\rightarrow \bigvee_{1 \le i \le k} \forall \bar{x}^{(n)} (P(\bar{x}^{(n)}) \leftrightarrow \Theta_i[\bar{x}^{(n)} \bar{v}^{(m)}]).$$

Theorem 4. If $\Sigma'(P) \cup \Gamma$ is such that in every model $\mathcal{A} \in M^t$ it is $|\{R : (\mathcal{A},R) \models \Sigma'\}| < \omega$ then the statement of the previous theorem is true#

Intuitively the above theorems /Theorems 3. and 4./ state the following: if $\Sigma'(P)$ is such that its required characteristics are fulfilled in every model by at least finite number of relations then there exists such a formula $\sigma \in F^t$ for calculating parameters $v_1, \ldots, v_m$ and there exist formulas $\Theta_1, \ldots, \Theta_k \in F^t$ out of v/hich one defines relation P by the parameters determined by σ

Fron the point of view of SAD this means that a theorem prover extended by induct-ive elements can prove, that

$$\Sigma'(P) \cup \Gamma \vdash \exists \bar{v}^{(m)} \sigma[\bar{v}^{(m)}].$$

On the basis of thin proof a zero-order termvector $\bar{\tau}^{(m)}$ muct bo selected so that $\Sigma' \models \sigma[\bar{\tau}^{(m)}]$. After this it has to be proved, that

$$\Sigma' \vdash \bigvee_{1 \le i \le k} \forall \bar{x}^{(n)} (P(\bar{x}^{(n)}) \leftrightarrow \Theta_i[\bar{x}^{(n)} \bar{\tau}^{(m)}]).$$

Then on the basis of knowledge Γ. we select the suitable defining formula $\Theta_i[\bar{\tau}^{(m)}]$ *

Now we further restrict the requirements concerning set of formulas $\Sigma'(P)$.

Theorem 5. If for each model $(\mathcal{A},R)$ of $\Sigma'(P) \cup \Gamma$ there exists such a finite $k \ge 1$, so $|\{R' : (\mathcal{A},R) \cong (\mathcal{A},R')\}| \le k$

then there exist such formulas $\sigma_j[\bar{v}^{(m)}]$, $\Theta_{ij}[\bar{x}^{(n)} \bar{v}^{(m)}]$ $(1 \le j \le r, 1 \le i \le k)$ in F^t that

$$\Sigma'(P) \cup \Gamma \models \bigvee_{1 \le j \le r} \exists \bar{v}^{(m)} \sigma_j[\bar{v}^{(m)}] \wedge \forall \bar{v}^{(m)} (\sigma_j[\bar{v}^{(m)}] \rightarrow$$

$$\rightarrow \bigvee_{1 \le i \le k} \forall \bar{x}^{(n)} (P(\bar{x}^{(n)}) \leftrightarrow \Theta_{ij}[\bar{x}^{(n)} \bar{v}^{(m)}])).$$

Theorem 6. /Kueker Theorem/: If $\Sigma'(P) \cup \Gamma$ is such that for each model $\mathcal{A} \in M^t$ there exists a finite $k \ge 1$, so

$$|\{R : (\mathcal{A},R) \models \Sigma'\}| \le k$$

then there exist such formulas $\sigma[\bar{v}^{(m)}]$,

$\Theta_i[\bar{x}^{(n)}\bar{G}^{(m)}]$ $(1 \le i \le k)$
in $_1F^t$ such that
$\Sigma^r(P) \cup \Gamma \models \exists \bar{G}^{(m)} \sigma[\bar{G}^{(m)}] \wedge \forall \bar{G}^{(m)}(\sigma[\bar{G}^{(m)}] \rightarrow$
$\rightarrow \bigwedge_{1 \le i \le k} \bar{x}^{(n)}(P(\bar{x}^{(n)}) \leftrightarrow \Theta_i[\bar{x}^{(n)}\bar{G}^{(m)}])$.

In these theorems similarly to Theorems 3. and 4. the formulas σ_j $(1 \le j \le r)$ and the formula σ serve to define the parametervector. The definition of relation 3 is done also on the basis of those described after Theorem 4- There is a difference only when definition is done on the basis of Theorem 5, because here we have to try out the formulas not only according t i $(1 \le i \le k)$ t also according to j $(1 \le j \le r)$.

The conditions of Theorems 5. and 6. for $\Sigma^r(P)$ are so much stronger than those of Theorems 3. and 4. that now we claim the existence of such a finite k which is upper-bound of the number of suitable relations in each model.

The $\Sigma^r(P)$ is t e strongest in that case if this conditions are satisfied in each model by at least one relation. Now we discuss those theorems which refer to this.

Theorem 7. /Svenonius' Theorem/: If for each model $(\mathcal{A}, R)$ of $\Sigma^r(P) \cup \Gamma$:
$|\{R' : (\mathcal{A},R) \cong (\mathcal{A},R')\}| \le 1$
then there exists a finite $m < \omega$ and there exist such formulas $\Theta_i[\bar{x}^{(m)}]$ $(1 \le i \le k)$ in F eo that,
$\Sigma^r(P) \cup \Gamma \models \bigvee_{1 \le i \le k} \forall \bar{x}^{(n)}(P(\bar{x}^{(n)}) \leftrightarrow \Theta_i[\bar{x}^{(n)}])$.

Intuitively it means that if we take two extensions $(\mathcal{A}, R_1)$ and $(\mathcal{A}, R_2)$ of any model $\mathcal{A} \in M^t$ so that these become models of $\Gamma \cup \Sigma(P)$ and these are isomorphic then $R_1 = R_2$.
In this case the set of formulas $\Sigma^r(P)$ defines relation P up to disjunction.

Theorem 8. /Beth'a Theorem/: If the set of formulas, $\Sigma^r(P) \cup \Gamma$ is such that for each model $\mathcal{A} \in M^t$
$|\{R : (\mathcal{A},R) \in \Sigma^r(P)\}| \le 1$
then there exists such a formula $\Theta[\bar{x}^{(n)}]$ in $_1F^t$ that
$\Sigma^r(P) \cup \Gamma \models \forall \bar{x}^{(n)}(P(\bar{x}^{(n)}) \leftrightarrow \Theta[\bar{x}^{(n)}])$.

Intuitively if $\Sigma^r(P)$ ia so strong that every model $\mathcal{A} \in M^t$ can be extended to a model o $\Sigma \cup \Gamma$ y at the most one relation then $\Sigma^r(P)$ defines relation *V* explicitly.

Conclusion

As we could see from aboves the model theory provides mathematical bases suitable for the development of different kinds of SAD important in the practice. This is expecially important because to construct implicitly described objects from psichological point of view is a task demanding creativity. The degree of creativity partly depends on the circumscription of the required object and partly on the development of the corresponding discipline. With the help of the theorems of different strength described in aboves we can obtain different SAD-s of different degree of creativity. So far we can see that the research of artificial intelligence requires the application of deep mathematical results of mathematical logic. To make SAD more effective we need the following problem to be solved: if $\Sigma^r(P) \subseteq_1 F^t$ and $\Gamma \subseteq_1 F^t$ are given then what conditions should $\Sigma^r(P)$ satisfy so as to have a formula $\Theta \in_1 F^t$ existing for which $\Gamma \models \Sigma(\Theta)$.

References

I.C.C.Chang, M.J.Keisler, Model Theory, Worth Holland, 1973-

2.G.D.Plotkin, A further note on inductive generalization, Machine Intelligence 6, Editors B. Meltzer, D.I.lichine, University PresB, Edinburgh, 1970.