SHARPENING THE CHARACTERIZATION OF THE POWER OF FLOYD METHOD

H. Andréka
Math. Inst. Hungar. Acad. Sci., Budapest
Reáltanoda u. 13-15, H-1053 Hungary

§1. INTRODUCTION

The present paper uses Nonstandard Dynamic Logic (from now on DL) to compare powers of program verification methods and to give explicit characterizations of well known program verification methods, such as e.g. Floyd's method. An exposition of this subject is [23] of which the present paper is a continuation. For more intuitive introductory and motivating material on the lattice of Dynamic Logics and on the lattice of program verification methods the reader is referred to [23].

The present paper is strongly related to [20] (in this volume) in several ways. E.g. both introduce the same Nonstandard DL, but [20] uses the earliest primitive version of Nonstandard DL in the form it was published in 1977 in [1]. The explicit characterization of the information content of Floyd method, which is refined in the present paper, was first published in [1] using the framework of [20]. Section 5 of [20] shows a modest one of the many applications of ultraproducts in the theory of programming. Another one is in the proof of Thm.6 of [23]. A third one is in the proof of Thm.2 of the present paper.

The first part (Section 2) of the present paper recalls the basic notions of Nonstandard DL since it is not as well known as one would like it to be. Hence the first part is rather similar to that of e.g. [23], [3], [4] etc. which may make the misleading impression on the reader that this might be the same paper. No, this isn't. Works belonging to Nonstandard DL are e.g. [4],[3],[27],[28],[1],[11],[9],[13],[20], [5],[14],[26]. A systematic introductory monograph with motivation, examples, overview of the field etc. is [4] which will be sent to anybody on request. A published introduction to Nonstandard DL with at least some of these features is [3]. Intuitive examples, illustrations are in [20],[22],[4]. One of the main ideas of Nonstandard DL is to restrict ourselves to logics with decidable proof concepts (see Def.9) while maintaining a reasonable, intuitively convincing model theory. We are interested in properties of program schemes <u>in axiomatizable classes of models</u>. We cannot understand why some people investigate properties of program schemes either in a single model or in the class of all possible models. Only these two extremes were treated e.g. in

the book of Manna.

By Thm.9 of [3], Floyd's method is equivalent with induction over formulas not containing quantifiers of sort time. Recently, E.M.Szabo raised the problem to decide how sharp this characterization is. In the present paper we prove that induction over formulas containing a single universal quantifier of sort time (i.e. Π_1-induction) is already strictly stronger than Floyd's method (see Thm.2 in §4).

NOTATION

In the following we shall recall some standard notations from textbooks on logic (mainly from [19],[8]).

d denotes an arbitrary similarity type of classical one-sorted models. I.e. d correlates arities (natural numbers) to function and relation symbols. See Def.1(i) in this paper.

ω denotes the set of natural nubers such that $0 \in \omega$.

Natural numbers are used in the von Neumann sense, i.e.

$n = \{0,1,\ldots,n-1\}$ and in particular

0 is the empty set.

$X = \{x_w : w \in \omega\}$ denotes a set of variables.

F_d is the set of classical first order formulas of type d with variables in X. Cf. e.g. [8]p.22.

τ denotes a term of type d in the usual sense of logic, see [8]p.22 or [19]p.166, Def.10.8(ii).

M_d denotes the class of all classical one-sorted models of type d, see e.g. [8] or [19] Def.11.1, or Defs. 1 and 3 here.

A classical one-sorted model is denoted by an underlined capital like $\underline{T}$ or $\underline{D}$ and its universe is denoted by the same capital without underlining. E.g. T is the universe of $\underline{T}$, and D is that of $\underline{D}$.

By a "valuation of the variables" in a model $\underline{D}$ a function $q : \omega \to D$ is understood, see [19]p.195.

$\tau[q]_{\underline{D}}$ denotes the value of the term τ in the model $\underline{D}$ under the valuation q of the variables, see [8]p.27, Def.13.13 or [19] Def.11.2. If τ contains no variable then we write τ instead of $\tau[q]_{\underline{D}}$, if $\underline{D}$ is understood.

$\underline{D} \models \varphi[q]$ denotes that the valuation q satisfies the formula φ in the model $\underline{D}$.

$L_d = \langle F_d, M_d, \models \rangle$ is the classical first order language of similarity type d, see [27].

^{A}B denotes the set of all functions from A into B, i.e. $^{A}B = \{f : f \text{ maps } A \text{ into } B\}$, see [19]p.7.

A function is considered to be a set of pairs.

Dom f denotes the domain of the function f, Dom $f \stackrel{d}{=} \{a : (\exists b)\langle a,b\rangle \in f\}$.

Rng f denotes the range of the function f, Rng $f \stackrel{d}{=} \{b : (\exists a)\langle a,b\rangle \in f\}$.

A sequence s of length n is a function with Dom $s = n$.

$\langle U_s : s \in S\rangle$ denotes the function $\{\langle s,U_s\rangle : s \in S\}$. Moreover for an expression Expr(x) and class S we define $\langle \text{Expr}(x) : x \in S\rangle$ to be the function $f : S \to \text{Rng } f$ such that $(\forall x \in S)$ $f(x) = \text{Expr}(x)$.

$\text{Sb}(X) \stackrel{d}{=} \{Y : Y \subseteq X\}$ is the powerset of X.

X^* denotes the set of all finite sequences of elements of X, i.e. $X^* \stackrel{d}{=} \cup\{^{m}X : m \in \omega\}$. We shall identify X^* with $\{H : H \subseteq X \text{ and } |H| < \omega\}^*$, and also with $(X^*)^*$. We think of X^* as the set of "words over the alphabet X".

$A \sim B \stackrel{d}{=} \{a \in A : a \notin B\}$.

§2. THE DEFINITION OF OUR DYNAMIC LOGIC DL_d.

2.1 Syntax of program schemes

Recall d, X, F_d from the list of notations. Now we define the set P_d of program schemes of type d.

The set Lab of "label symbols" is defined to be an arbitrary but fixed subset of the set Tm_d^0 of all constant terms of type d, i.e. d-type terms which do not contain variable symbols. (Lab is chosen this way for technical reasons only. There are many other possible ways for handling labels, see e.g. [28].) Logical symbols: $\{\wedge, \neg, \exists, =\}$. Other symbols: $\{\leftarrow, \text{IF}, \text{GOTO}, \text{HALT}, (,), :\}$.

The set U_d of commands of type d is defined as follows:

$(i : x \leftarrow \tau) \in U_d$ if $i \in \text{Lab}$, $x \in X$, and τ is a term of type d and with all variables in X.

$(i : \text{IF } \chi \text{ GOTO } v) \in U_d$ if $i, v \in \text{Lab}$, $\chi \in F_d$ is a formula without quantifier.

$(i : \text{HALT}) \in U_d$ if $i \in \text{Lab}$.

These are the only elements of U_d.

By a program scheme of type d we understand a finite sequence p of commands (elements of U_d) ending with a "HALT", in which no two members have the same label, and in which the only "HALT-command" is

the last one. Further, if $(i: \text{IF } \chi \text{ GOTO } v)$ occurs in p then there is u such that the command $(v:u)$ occurs in p. I.e. an element p of P_d is of the form $p = \langle (i_0:u_0),\dots,(i_{n-1}:u_{n-1}), (i_n:\text{HALT})\rangle$ where $n\in\omega$, $(i_m:u_m)\in U_d$ for $m\leq n$ etc.

Convention 1 If a program scheme is denoted by p then its parts are denoted as follows:

$p = \langle(i_0:u_0),\dots,(i_{n-1}:u_{n-1}),(i_n:\text{HALT})\rangle$.

Throughout we shall use the definition:

$c \overset{d}{=} \min\{w\in\omega : (\forall v\in\omega\sim w)[x_v \text{ does not occur in } p]\}$.

I.e. $\{x_w : w<c\}$ contains all the variables occurring in the program scheme p, and if $c>0$ then x_{c-1} really occurs in p. We shall use x_c as the control variable of p.

2.2 Semantics of program schemes

By a language with semantics we understand a triple $L = \langle F , M , \models\rangle$ of classes such that $\models \subseteq M\times F\times Sets$ where Sets is the class of all sets. Here F is called the syntax of L, M the class of models or possible interpretations of L, $\models$ the satisfaction relation of L, and $\langle M, \models\rangle$ is called the semantics of L. Instead of $\langle a,b,c\rangle\in\models$ we write $a \models b[c]$, and we say "c satisfies b in a". See [27],[29] p.265.

DEFINITION 1 (one-sorted models)

(i) By a (classical or one-sorted) similarity type d we understand a pair $d = \langle H,d_1\rangle$ such that d_1 is a function $d_1 : \sum \to \omega$ for some set $\sum$, $H\subseteq\sum$ and $(\forall r\in\sum)d_1(r)>0$.

The elements of $\sum$ are called the symbols of d and the elements of H are called the operation symbols or function symbols of d. Let $r\in\sum$. Then we shall write $d(r)$ instead of $d_1(r)$.

(ii) Let $d = \langle H,d_1\rangle$ be a similarity type, let $\sum = \text{Dom } d_1$ as above. By a model of type d we understand a pair $\underline{D} = \langle D,R\rangle$ such that R is a function with $\text{Dom } R = \sum$ and $(\forall r\in\sum)R(r)\subseteq {}^{d(r)}D$ and if $r\in H$ then $R(r) : {}^{(d(r)-1)}D \to D$.

Notation: $\langle D,R_r\rangle_{r\in\sum} \overset{d}{=} \langle D,\langle R_r : r\in\sum\rangle\rangle \overset{d}{=} \langle D,R\rangle$.

I.e. $\underline{D} = \langle D,R_r\rangle_{r\in\sum}$ is a model of type d iff R_r is a $d(r)$-ary

relation over D and if $r\in H$ then R_r is a $(d(r)-1)$-ary function, for all $r\in\sum$.

If $r\in H$ and $d(r)=1$ then there is a unique $b\in D$ such that $R_r = \{\langle b\rangle\}$ and we shall identify R_r with b. If $r\in H$, $d(r)=1$ then r is said to be a constant symbol and $R_r\in D$ is the constant element denoted by r in $\mathfrak{Q}$.

The set D is called the universe of $\mathfrak{Q}$.

(iii) $M_d \overset{d}{=} \{\mathfrak{Q} : \mathfrak{Q}$ is a model of type $d\}$. End of Definition 1

DEFINITION 2 (the similarity type t of arithmetic)

t denotes the similarity type of Peano's arithmetic. In more detail, $t = \langle \{0,sc,+,\cdot\}, t_1\rangle$ where $\mathrm{Dom}\, t_1 = \{\leq,0,sc,+,\cdot\}$, $t(\leq)=2$, $t(0)=1$, $t(sc)=2$ and $t(+)=t(\cdot)=3$. End of Definition 2

Throughout the paper, t is supposed to be disjoint from any other similarity type, moreover if d is a similarity type then $\mathrm{Dom}(d_1)\cap\mathrm{Dom}(t_1)=0$ is assumed throughout the paper.

DEFINITION 3 (many-sorted models, [19])

(i) By a many-sorted similarity type m we understand a triple $m = \langle S,H,m_2\rangle$ such that m_2 is a function $m_2 : \sum \to S^*$ for some set $\sum$, $H\subseteq\sum$ and $(\forall r\in\sum)m_2(r) \neq {}^0S$.

The elements of S are called the sorts of m. If $r\in\sum$ then we shall write $m(r)$ instead of $m_2(r)$.

(ii) Let m be a many-sorted similarity type and let $\sum = \mathrm{Dom}\, m_2$ as above. By a (many-sorted) model of type m we understand a pair $\mathfrak{M} = \langle\langle U_s : s\in S\rangle, R\rangle$ such that R is a function with $\mathrm{Dom}\, R = \sum$ and if $r\in\sum$ and $m(r)=\langle s_1,\ldots,s_n\rangle$ then $R(r)\subseteq U_{s_1}\times\ldots\times U_{s_n}$ and if in addition $r\in H$ then $R(r)$ is a function $R(r) : U_{s_1}\times\ldots\times U_{s_{n-1}} \to U_{s_n}$.

U_s is said to be the universe of sort s of $\mathfrak{M}$.

(iii) $M_m \overset{d}{=} \{\mathfrak{M} : \mathfrak{M}$ is a many-sorted model of type $m\}$.

End of Definition 3

DEFINITION 4 (the 3-sorted similarity type td)

(i) To any one-sorted similarity type d we associate a 3-sorted similarity type td as follows:

Let $d = \langle H,d_1\rangle$ be any one-sorted similarity type. Recall that t is a fixed similarity type introduced in Def.2 and, by our convention, $\mathrm{Dom}(d_1)\cap\mathrm{Dom}(t_1)=0$.

We shall define the new similarity type td such that the original similarity types t and d will be actual elements of a part of td, namely t and d will be two sorts of td.

Now we define td to be $td \stackrel{d}{=} \langle S, K, td_2 \rangle$ where

a) $S \stackrel{d}{=} \{t,d,i\}$, $|S|=3$. (S is the set of sorts of td.) Here the elements of S are used as symbols only; we could have chosen $S = \{0,1,2\}$ as well.

b) $K \stackrel{d}{=} \{ext, 0, sc, +, \cdot\} \cup H$. (K is the set of operation symbols of td.)

c) $td_2 : (Dom(t_1) \cup Dom(d_1) \cup \{ext\}) \to S^*$ such that
$td_2(ext) = \langle i,t,d \rangle$,
$td_2(r) \in^n \{t\}$ if $t(r)=n$ and
$td_2(r) \in^n \{d\}$ if $d(r)=n$.
E.g. $td_2(\leq) = \langle t,t \rangle$, $td_2(+) = \langle t,t,t \rangle$, etc.

By these the 3-sorted similarity type td is defined.

(ii) Let $\mathfrak{M} = \langle \langle U_t, U_d, U_i \rangle, R_r \rangle_{r \in \sum}$ be a td-type model. Then (1)--(3) below hold:

(1) $\langle U_t, R_r \rangle_{r \in Dom(t_1)} \in M_t$.

(2) $\langle U_d, R_r \rangle_{r \in Dom(d_1)} \in M_d$.

(3) $R_{ext} : U_i \times U_t \to U_d$.

Notation: $\langle \langle U_t, R_r \rangle_{r \in Dom(t_1)}, \langle U_d, R_r \rangle_{r \in Dom(d_1)}, U_i, R_{ext} \rangle \stackrel{d}{=}$
$\stackrel{d}{=} \langle \langle U_t, U_d, U_i \rangle, R_r \rangle_{r \in \sum}$.

We define: $\underset{\sim}{T} \stackrel{d}{=} \langle U_t, R_r \rangle_{r \in Dom(t_1)}$, $T \stackrel{d}{=} U_t$,

$\underset{\sim}{D} \stackrel{d}{=} \langle U_d, R_r \rangle_{r \in Dom(d_1)}$, $D \stackrel{d}{=} U_d$ and $I \stackrel{d}{=} U_i$.

The sorts t,d, and i are called time, data and intensions respectively. $\underset{\sim}{T}$ is said to be the time-structure of $\mathfrak{M}$.

End of Definition 4

Convention 2 Whenever an element of M_{td} is denoted by the letter $\mathfrak{M}$ then the parts of $\mathfrak{M}$ are denoted as follows:

$\langle \underset{\sim}{T}, \underset{\sim}{D}, I, ext \rangle \stackrel{d}{=} \langle \langle U_t^{\mathfrak{M}}, U_d^{\mathfrak{M}}, U_i^{\mathfrak{M}} \rangle, r^{\mathfrak{M}} \rangle_{r \in \sum} \stackrel{d}{=} \mathfrak{M}$.

Note that $\mathfrak{M} \in M_{td}$ iff $[\underset{\sim}{T} \in M_t$, $\underset{\sim}{D} \in M_d$, and $ext : I \times T \to D]$.

For more detailed introduction to many-sorted languages, like $L_{td} = \langle F_{td}, M_{td}, \models \rangle$ defined below, the reader is referred e.g. to the textbook [19].

DEFINITION 5 (the first order 3-sorted language $L_{td} = \langle F_{td}, M_{td}, \models \rangle$ of type td, [19])

Let $d = \langle H, d_1 \rangle$ be any one-sorted similarity type. Recall from Def.s 3 and 4 that t is a fixed similarity type, and td is a 3-sorted similarity type with sorts $\{t,d,i\}$.

(i) We define the set F_{td} of first order 3-sorted formulas of type td.:

Let $X \stackrel{d}{=} \{x_w : w\in\omega\}$, $Y \stackrel{d}{=} \{y_w : w\in\omega\}$ and $Z \stackrel{d}{=} \{z_w : w\in\omega\}$ be three disjoint sets (and $x_w \neq x_j$ if $w\neq j\in\omega$ etc.). We define Z, X, and Y to be the sets of variables of sorts t, d, and i respectively.

F_t^Z denotes the set of all first order formulas of type t with variables in Z, F_d denotes the set of all first order formulas of type d with variables in X, and Tm_t^Z denotes the set of all first order terms of type t with variables in Z.

The set $Tm_{td,d}$ of terms of type td and of sort d is defined to be the smallest set satisfying conditions (1)-(3) below.

(1) $X \subseteq Tm_{td,d}$.

(2) $ext(y_w,\tau)\in Tm_{td,d}$ for any $\tau\in Tm_t^Z$ and $w\in\omega$.

(3) $f(\tau_1,\ldots,\tau_n)\in Tm_{td,d}$ for any $f\in H$ if $d(f)=n+1$ and $\tau_1,\ldots,\tau_n \in Tm_{td,d}$.

The set F_{td} of first order formulas of type td is defined to be the smallest set satisfying conditions (4)-(8) below.

(4) $(\tau_1=\tau_2)\in F_{td}$ for any $\tau_1,\tau_2\in Tm_{td,d}$.

(5) $r(\tau_1,\ldots,\tau_n)\in F_{td}$ for any $\tau_1,\ldots,\tau_n\in Tm_{td,d}$ and for any $r\in H$ if $d(r)=n$.

(6) $(y_w=y_j)\in F_{td}$ for any $w,j\in\omega$.

(7) $F_t^Z \subseteq F_{td}$.

(8) $\{\neg\varphi, (\varphi\wedge\psi), (\exists x_w\varphi), (\exists y_w\varphi), (\exists z_w\varphi) : w\in\omega\} \subseteq F_{td}$ for any $\varphi,\psi\in F_{td}$.

By this, the set F_{td} has been defined. Note that $F_d \subseteq F_{td}$.

(ii) Now we define the "meanings" of elements of F_{td}.

By a valuation (of the variables) into $\mathfrak{M}$ we understand a triple $v=\langle g,k,r\rangle$ such that $g \in {}^{\omega}T$, $k \in {}^{\omega}D$ and $r \in {}^{\omega}I$. The statement "the valuation $v=\langle g,k,r\rangle$ satisfies φ in $\mathfrak{M}$" is denoted by $\mathfrak{M} \models \varphi[v]$ or equivalently by $\mathfrak{M} \models \varphi[g,k,r]$.

The truth of $\mathfrak{M} \models \varphi[g,k,r]$ is defined the usual way(see [1g]) which is completely analogous with the one-sorted case. E.g.

$\mathfrak{M} \models (y_0=y_1)[g,k,r]$ iff $r_0=r_1$,

$\mathfrak{M} \models (x_1=ext(y_2,z_0))[g,k,r]$ iff $k_1=ext^{\mathfrak{M}}(r_2,g_0)$,

$\mathfrak{M} \models \varphi[g,k,r]$ iff $\underset{\sim}{T} \models \varphi[g]$ for $\varphi\in F_t^Z$,

$\mathfrak{M} \models \varphi[g,k,r]$ iff $\underset{\sim}{D} \models \varphi[k]$ for $\varphi \in F_d$ etc.

The formula $\varphi \in F_{td}$ is <u>valid</u> in $\mathfrak{M}$, in symbols $\mathfrak{M} \models \varphi$, iff

$(\forall g \in {}^{\omega}T)(\forall k \in {}^{\omega}D)(\forall r \in {}^{\omega}I)\ \mathfrak{M} \models \varphi[g,k,r]$.

(iii) The (3-sorted) language L_{td} of type td is defined to be the triple $L_{td} = \langle F_{td}, M_{td}, \models \rangle$ where $\models$ is the satisfaction relation defined in (ii) above. <u>End of Definition 5</u>

Now we define the <u>meanings of program schemes</u> $p \in P_d$ in the 3-sorted models $\mathfrak{M} \in M_{td}$.

<u>Notation:</u> Let $\langle \underset{\sim}{T}, \underset{\sim}{D}, I, ext \rangle \in M_{td}$, see Convention 2. Let $s_0, \ldots, s_m \in I$, $\bar{s} \overset{d}{=} \langle s_0, \ldots, s_m \rangle$. Let $b \in T$. Then we define

$ext(\bar{s},b) \overset{d}{=} \langle ext(s_0,b), \ldots, ext(s_m,b) \rangle$.

<u>DEFINITION 6</u> (traces of programs in time-models)

Let $p \in P_d$ and $\mathfrak{M} \in M_{td}$. We shall use Conventions 1 and 2. Let $s_0, \ldots, s_c \in I$ be arbitrary intensions in $\mathfrak{M}$. Let $\bar{s} = \langle s_0, \ldots, s_{c-1} \rangle$. The sequence $\langle s_0, \ldots, s_c \rangle$ of intensions is defined to be a <u>trace of p in $\mathfrak{M}$</u> if the following (i) and (ii) are satisfied.

(i) $ext(s_c,0) = i_0$ and $ext(s_c,b) \in \{i_m : m \leq n\}$ for every $b \in T$.

(ii) For every $b \in T$ and for every $j \leq c$ if $ext(s_c,b) = i_m$ then statements (1)-(3) below hold.

(1) If $u_m =$ "$x_w \leftarrow \tau$" then

$$ext(s_j,b+1) = \begin{cases} i_{m+1} & \text{if } j \neq w \\ \tau[ext(\bar{s},b)]_{\underset{\sim}{D}} & \text{if } j = w \\ ext(s_j,b) & \text{otherwise} \end{cases}$$

(2) If $u_m =$ "IF χ GOTO v" then

$$ext(s_j,b+1) = \begin{cases} v & \text{if } j=c \text{ and } \underset{\sim}{D} \models \chi[ext(\bar{s},b)] \\ i_{m+1} & \text{if } j=c \text{ and } \underset{\sim}{D} \not\models \chi[ext(\bar{s},b)]. \\ ext(s_j,b) & \text{otherwise} \end{cases}$$

(3) If $u_m =$ "HALT" then $ext(s_j,b+1) = ext(s_j,b)$.

<u>End of Definition 6</u>

<u>DEFINITION 7</u> (possible output)

Let $s = \langle s_0, \ldots, s_c \rangle$ be a trace of $p \in P_d$ in $\mathfrak{M} \in M_{td}$.

(i) Let $k \in {}^{\omega}D$. The trace s is said to be <u>of input</u> k iff $(\forall j<c)k(j) = ext(s_j,0)$.

(ii) Recall from Convention 1 that i_n is the label of the HALT-command of p. Let $b \in T$. We say that <u>s terminates p at time b</u> in $\mathfrak{M}$ iff $ext(s_c,b)=i_n$.

(iii) Let $k,q \in {}^{\omega}D$. We define q to be a <u>possible output</u> of p <u>with input</u> k in $\mathfrak{M}$ iff (a)-(d) below hold for some s.

(a) $s = \langle s_0,\ldots,s_c \rangle$ is a trace of p in $\mathfrak{M}$.

(b) s is of input k.

(c) There is $b \in T$ such that s terminates p at time b and $\langle q_0,\ldots,q_{c-1} \rangle = \langle ext(s_0,b),\ldots,ext(s_{c-1},b) \rangle$.

(d) $(\forall j \in \omega)[j \geq c \Rightarrow q_j=k_j]$.

If q is a possible output of p with input k in $\mathfrak{M}$ then we shall also say that $\langle q_0,\ldots,q_{c-1} \rangle$ is a possible output of p with input $\langle k_0,\ldots,k_{c-1} \rangle$. <u>End of Definition 7</u>

By now we have defined a semantics of program schemes.

<u>Remark:</u> A trace $\langle s_0,\ldots,s_c \rangle$ of a program $p \in P_d$ correlates to each variable x_w ($w \leq c$) occurring in the program p an intension or "<u>history</u>" s_w such that the value $ext(s_w,b)$ can be considered as the "value contained in" or "extension of" x_w at time point $b \in T$. The intension $s_w \in I$ represents a function $ext(s_w,-) : T \to D$ from time points of data values D. This function is the "history" of the variable x_w during an <u>execution of the program p in the model $\mathfrak{M}$</u>. Def.6 ensures that the sequence $\langle ext(s_0,-),\ldots,ext(s_c,-) \rangle$ of functions can be considered as a behaviour or "run" or "trace" of the program p in $\mathfrak{M}$. Here s_c is the intension of the "control variable".

<u>About using Th.:</u> It might look counter-intuitive to execute programs in arbitrary elements of M_{td}. However, we can collect <u>all our postulates about time</u> into a set $Ax \subseteq F_{td}$ of axioms which this way would define the class $Mod(Ax) \subseteq M_{td}$ of all <u>intended interpretations</u> of P_d. Then traces of programs in $Mod(Ax)$ provide an intuitively acceptable semantics of program schemes. Such a set Ax of axioms was proposed in [23] Def.13 and in [3] Part II, Def.14. <u>If</u> one wants to define semantics with unusual time structure e.g. parallelism, nondeterminism, interactions etc. then one can choose an Ax <u>different</u> from the one proposed in [23] or [3].

2.3. The language DL_d for reasoning about programs

We introduce our language DL_d for reasoning about programs or, in other words, the language DL_d of our first order dynamic logic.

DEFINITION 8 (the language DL_d of first order dynamic logic)

Let d be a (one-sorted) similarity type.

(i) DF_d is defined to be the smallest set satisfying conditions (1)-(3) below.

(1) $F_{td} \subseteq DF_d$.

(2) $(\forall p \in P_d)(\forall \varphi \in DF_d) \Box(p,\varphi) \in DF_d$.

(3) $(\forall \varphi,\psi \in DF_d)(\forall x \in X \cup Y \cup Z)\{\neg\varphi,\ (\varphi \wedge \psi),\ (\exists x \varphi)\} \subseteq DF_d$.

By this we have defined the set DF_d of dynamic formulas of type d.

(ii) Now we define the meanings of the dynamic formulas in the 3-sorted models $\mathfrak{M} \in M_{td}$. Let $\mathfrak{M} = \langle \underset{\sim}{T}, \underset{\sim}{D}, I, ext \rangle \in M_{td}$. Let v be a valuation of the variables of F_{td} into $\mathfrak{M}$, i.e. let $v = \langle g,k,r \rangle$ where $g \in {}^{\omega}T$, $k \in {}^{\omega}D$, and $r \in {}^{\omega}I$. We shall define the truth of $\mathfrak{M} \models \varphi[v]$ for all $\varphi \in DF_d$.

(4) If $\varphi \in F_{td}$ then $\mathfrak{M} \models \varphi[v]$ is already defined in Def.5.

(5) Let $p \in P_d$ and $\psi \in DF_d$ be arbitrary. Assume that $\mathfrak{M} \models \psi[v]$ has already been defined for every valuation v of the variables of F_{td} into $\mathfrak{M}$. Let $g \in {}^{\omega}T$, $k \in {}^{\omega}D$, and $r \in {}^{\omega}I$. Then $\mathfrak{M} \models \Box(p,\psi)[g,k,r]$ iff $[\mathfrak{M} \models \psi[g,q,r]$ for every possible output q of p with input k in $\mathfrak{M}]$. For "possible output" see Def.7.

(6) Let $\varphi,\psi \in DF_d$ and let $x \in X \cup Y \cup Z$. Then $\mathfrak{M} \models (\neg\varphi)[g,k,r]$, $\mathfrak{M} \models (\varphi \wedge \psi)[g,k,r]$ and $\mathfrak{M} \models (\exists x \varphi)[g,k,r]$ are defined the usual way.

Let e.g. $w \in \omega$. Then $\mathfrak{M} \models (\exists z_w \varphi)[g,k,r]$ iff (there is $h \in {}^{\omega}T$ such that $(\forall j \in \omega)(j \neq w \Rightarrow h_j = g_j)$ and $\mathfrak{M} \models \varphi[h,k,r]$).

(iii) The language DL_d of first order dynamic logic of type d is defined to be the triple $DL_d \overset{d}{=} \langle DF_d, M_{td}, \models \rangle$ where $\models$ is defined in (ii) above. End of Definition 8

Notation: Let $p \in P_d$ and $\psi \in DF_d$. Then $\Diamond(p,\psi)$ abbreviates the formula $\neg\Box(p,\neg\psi)$. In our language DF_d we introduced the logical connectives $\neg,\wedge,=,\exists,\Box$ only. However, we shall use the derived logical connectives $\forall,\rightarrow,\leftrightarrow,\vee$, TRUE, FALSE, $\Diamond$, too, in the standard sense.

E.g. $(\varphi \vee \psi)$ stands for the formula $\neg(\neg\varphi \wedge \neg\psi)$.

Remark: Standard concepts of programming theory can be expressed in DL_d. E.g. $\Box(p,\psi)$ expresses that p is partially correct w.r.t. output condition ψ, and $\Diamond(p,\psi)$ expresses that p is totally correct w.r.t. output condition ψ in the weaker sense.

Convention 3 We shall use the model theoretic consequence relation $\models$ in the usual way. I.e. Let $Th \subseteq DF_d$, $\psi \in DF_d$ and $K \subseteq M_{td}$. Then

$\mathfrak{M} \models \varphi$ iff $(\forall g \in {}^{\omega}T)(\forall k \in {}^{\omega}D)(\forall r \in {}^{\omega}I)\ \mathfrak{M} \models \varphi[g,k,r]$,

$\mathfrak{M} \models Th$ iff $(\forall \varphi \in Th)\ \mathfrak{M} \models \varphi$,

$K \models Th$ iff $(\forall \mathfrak{M} \in K)\ \mathfrak{M} \models Th$,

$Mod(Th) \overset{d}{=} Mod_{td}(Th) \overset{d}{=} \{\mathfrak{M} \in M_{td} : \mathfrak{M} \models Th\}$, and

$Th \models \varphi$ iff $Mod(Th) \models \varphi$.

Note that $Mod(Th)$ is a sloppy abbreviation of $Mod_{td}(Th)$, we shall use it when context helps the reader to guess which similarity type h such that $Th \subseteq F_h$ is used in $Mod(Th) = Mod_h(Th)$.

2.4. Proof concepts

DEFINITION 9 (proof concept [19])

Let $L = \langle F, M, \models \rangle$ be a language. By a proof concept on the set F we understand a relation $\vdash \subseteq Sb(F) \times F$ together with a set $Pr \subseteq F^*$ such that $(\forall Th \subseteq F)(\forall \varphi \in F)[Th \vdash \varphi$ iff $(\langle H, w, \varphi \rangle \in Pr$ for some finite $H \subseteq Th$ and for some $w \in F^*)]$. Recall that we identify X^* with $\{H \in Sb(X) : |H| < \omega\}^*$.

The proof concept $(\vdash, Pr)$ is decidable iff the set Pr is a decidable subset of F^* in the usual sense of the theory of algorithms and recursive functions (i.e. if Pr is recursive).

Pr is called the set of proofs, and $\vdash$ is called derivability relation. End of Definition 9

Sometimes we shall sloppily write "$\vdash$ is a decidable proof concept" instead of "$(\vdash, Pr)$ is a decidable proof concept".

Note that the usual proof concept of classical first order logic is a decidable one in the sense of the above definition. As a contrast we note that the so called effective ω-rule is not a decidable proof concept.

THEOREM 1 (strong completeness of DL_d)

There is a decidable proof concept $(\vdash^N, Prn)$ for the language DL_d such that for every $Th \subseteq DF_d$ and $\varphi \in DF_d$ we have $[Th \models \varphi$ iff $Th \vdash^N \varphi]$.

Proof: can be found in [3], as well as in [4] Thm.2 pp.30-38. QED

DEFINITION 10 (the proof concept $(\vdash^N, Prn)$ of DL_d)

By Thm.1 above there exists a decidable set $Prn \subseteq (DF_d)^*$ such that $(\forall Th \subseteq DF_d)(\forall \varphi \in DF_d)[Th \vdash^N \varphi$ iff $(\exists$ finite $H \subseteq Th)(\exists w)\langle H,w,\varphi\rangle \in Prn]$.

The decision algorithm for Prn is rigorously constructed in [3] Thm.2, and [4]Thm.2, pp.30-38, and in [21].

From now on we shall use Prn as defined in the quoted papers. The only important properties of Prn we shall use are its decidability and its completeness for DL_d. End of Definition 10

DEFINITION 11 (new proof concepts $(Ax \vdash^N)$ from $\vdash^N$)

Let $Ax \subseteq DF_d$ be decidable but otherwise arbitrary.

(i) Let $Th \subseteq DF_d$ and $\varphi \in DF_d$ be arbitrary. We say that φ is $(Ax \vdash^N)$-provable from Th iff $Th \cup Ax \vdash^N \varphi$. That is, φ is provable by the proof concept $(Ax \vdash^N)$ from Th iff $Th \cup Ax \vdash^N \varphi$. Thus $(Ax \vdash^N)$ is a new recursively enumerable "provability" relation.

(ii) $pf(Ax \vdash^N) \overset{d}{=} \{\langle H,\langle L,w\rangle,\varphi\rangle \in (DF_d)^* : \langle HUL,w,\varphi\rangle \in Prn$ and $L \subseteq Ax\}$. Clearly φ is $(Ax \vdash^N)$-provable from Th iff $(\exists \langle H,w,\varphi\rangle \in Prn) H \subseteq Th \cup Ax$. Clearly $pf(Ax \vdash^N)$ is a decidable subset of $(DF_d)^*$.

(iii) We have defined a new proof concept $\langle (Ax \vdash^N), pf(Ax \vdash^N)\rangle$ where $pf(Ax \vdash^N)$ is the decidable set of all $(Ax \vdash^N)$-proofs. We shall always denote this new proof concept by $(Ax \vdash^N)$. So whenever we write $(Ax \vdash^N)$ we shall mean $\langle (Ax \vdash^N), pf(Ax \vdash^N)\rangle$ but we shall not write it out explicitly. End of Definition 11

§3. THE LATTICE OF PROOF METHODS FOR PARTIAL CORRECTNESS OF PROGRAMS

In [23], it was shown how to use the logic DL_d to compare powers of methods of program verification as well as to generate new methods for program verification. To this end, we had to fix the criteria to be used when we compare program verification methods. We say that one method $\vdash_1$ is stronger than another $\vdash_2$ iff more programs can be proved to be partially correct by $\vdash_1$ than by $\vdash_2$. So we consider the reasoning power to prove partial correctness statements $\varphi \to \Box(p,\psi)$ to be the criterion to compare different

methods. This choice has nothing to do with our logic DL_d, namely DL_d is suitable for proving total correctness of programs. It was proved in [3]Thm.7 and in Thm.7 of [4] that the Kfoury-Park [16] negative result on proving total correctness is not true for DL_d.

We shall consider program verification methods only with decidable proof concepts.

On Figure 2 of [23], different proof concepts $(Ax_1 \vdash^N)$, $(Ax_2 \vdash^N)$ were compared with each other as well as with such classical proof concepts as Floyd's $\vdash^F$, Burstall's $\vdash^{mod}$ and Pnueli's $\vdash^{fum}$. Intuitive motivations for investigating Fig.2 can be found in [23] §5.2. Below we recall Fig.2 of [23] (see Fig.1 here) together with the most important definitions.

DEFINITION 12 (ind(φ,z), IA, Ia, Lax)

Let d be a similarity type. Then td, F_{td} and Z were defined in Def.s 4 and 5 in §2.2. Let $z \in Z$ be arbitrary. Let $\varphi \in F_{td}$. We define the induction formula ind(φ,z) as follows:

$$ind(\varphi,z) \stackrel{d}{=} ([\varphi(0) \wedge \forall z(\varphi \to \varphi(sc(z))] \to \forall z\varphi),$$

where $\varphi(0)$ and $\varphi(sc(z))$ denote the formulas obtained from φ by replacing every free occurrence of z in φ by 0 and sc(z) resp.

The induction axioms are:

$$IA \stackrel{d}{=} \{ind(\varphi,z) : \varphi \in F_{td} \text{ and } z \in Z\}.$$

$$Lax \stackrel{d}{=} \{(j \neq k) : j \text{ and } k \text{ are two different elements of Lab}\}.$$

$$Ia \stackrel{d}{=} IA \cup Lax.$$

End of Definition 12

Clearly $IA \subseteq F_{td}$ since if $\varphi \in F_{td}$ and $z \in Z$ then $\varphi(0)$, $\varphi(sc(z)) \in F_{td}$ because 0 and sc(z) are terms of sort t. It is important to stress here that φ may contain other free variables of all sorts. All the free variables of φ are also free in ind(φ,z) except for z. They are the "parameters" of the induction ind(φ,z).

The theory IA says that if a "property" φ changes during time T then it must change "some time", i.e. there is a time point $b \in T$ when φ is just changing.

Our strongest set of induction axioms is Ia. We shall distinguish various subsets of Ia.

DEFINITION 13 (Iq, $I\Sigma_1$, $I\Pi_1$, Il, Ict, If)

$$If \stackrel{d}{=} \{\varphi \in IA : \varphi \text{ contains no free variable of sort t or d}\} \cup Lax.$$

$$Il \stackrel{d}{=} \{\varphi \in IA : (\forall i \in \omega)[i>0 \Rightarrow z_1 \text{ does not occur in } \varphi \text{ neither free}$$

nor bound}∪Lax.

$Ict \stackrel{d}{=} \{ind(\exists x_0 \ldots x_m[(\bigwedge_{j\leq m} x_j = ext(y_j, z_0)) \wedge \varphi], z_0) : m\in\omega \text{ and } \varphi\in F_d\} \cup Lax.$

$(\sum_{0,t} F_{td}) \stackrel{d}{=} \{\varphi\in F_{td} :$ φ contains no quantifier of sort t, that is $(\forall i\in\omega)[\text{"}\exists z_i\text{"}$ does not occur in $\varphi]\}$.

$Iq \stackrel{d}{=} \{ind(\varphi, z_0) : \varphi\in(\sum_{0,t} F_{td})\} \cup Lax.$

$I\Sigma_1 \stackrel{d}{=} \{ind(\exists z_1 \ldots z_m \varphi, z_0) : \varphi\in(\sum_{0,t} F_{td}) \text{ and } m\in\omega\} \cup Lax.$

$I\Pi_1 \stackrel{d}{=} \{ind(\forall z_1 \ldots z_m \varphi, z_0) : \varphi\in(\sum_{0,t} F_{td}) \text{ and } m\in\omega\} \cup Lax.$

End of Definition 13

DEFINITION 14 (Ts, To, Tpres, $Tpa \subseteq F_t^z$)

Notation: $sc^0(z_0) \stackrel{d}{=} z_0$ and $(\forall n\in\omega) sc^{n+1}(z_0) \stackrel{d}{=} sc(sc^n(z_0))$.

$Ts \stackrel{d}{=} \{z_0 \neq 0 \leftrightarrow \exists z_1(z_0 = sc(z_1)),\ sc(z_0)=sc(z_1) \rightarrow z_0=z_1,\ sc^n(z_0)\neq z_0 :$
$: n\in\omega,\ n\neq 0\}$.

$To \stackrel{d}{=} \{(z_0\leq z_1 \wedge z_1\leq z_2) \rightarrow z_0\leq z_2,\ (z_0\leq z_1 \wedge z_1\leq z_0) \rightarrow z_0=z_1,$
$z_0\leq z_1 \vee z_1\leq z_0,\ 0\leq z_0,\ (z_0\leq z_1 \wedge z_0\neq z_1) \leftrightarrow sc(z_0)\leq z_1,$
$0=z_0 \vee z_1(z_0=sc(z_1))\}$.

Tpres is the deicable set of Presburger's axioms:

$Tpres \stackrel{d}{=} To \cup \{z_0+0=z_0,\ z_0+sc(z_1)=sc(z_0+z_1),\ ind(\varphi, z_0) : \varphi\in F_t^z$ and
"." does not occur in $\varphi\}$.

Tpa is the set of Peano's axioms formulated in the language F_t^z about the similarity type t, see e.g. Example 1.4.11 in [8]p.42.:

$Tpa \stackrel{d}{=} Tpres \cup \{z_0\cdot 0=0,\ z_0\cdot sc(z_1)=(z_0\cdot z_1)+z_1,\ ind(\varphi, z_0) : \varphi\in F_t^z\}$.

End of Definition 14

DEFINITION 15 (Floyd-Hoare proof method $(\vdash^F, Prf)$)

(i) The set HF_d of Floyd-Hoare statements of type d is an important sublanguage of DF_d:

$HF_d \stackrel{d}{=} \{(\varphi \rightarrow \Box(p,\psi)) : p\in P_d \text{ and } \varphi,\psi\in F_d\}$. Clearly $HF_d \subseteq DF_d$.

(ii) Floyd-Hoare language HFL_d is defined to be:

$HFL_d \stackrel{d}{=} \langle HF_d \cup F_d,\ Mod_{td}(Iq),\ \models\rangle$.

(iii) The relation $\vdash^F \subseteq \{Th : Th\subseteq F_d\}\times HF_d$ was defined in a rigorous manner in [20]Def.4 (this volume), [3]Def.17, [4]Def.17 p.55, [6],[2]p.118. We shall use this definition of $\vdash^F$ without refor -

-mulating it, but we note that in the quoted papers there is a decidable set $Prf \subseteq (HF_d \cup F_d)^*$ such that $(\forall Th \subseteq F_d)(\forall \varphi \in HF_d)[Th \vdash^F \varphi$ iff $(\exists$ finite $H \subseteq Th)(\exists w)\langle H,w,\varphi\rangle \in Prf]$. Hence Prf is the set of $\vdash^F$-proofs and Prf is decidable. Cf. Def.10. According to Def.10, $(\vdash^F, Prf)$ is a decidable proof concept for the Floyd-Hoare language HFL_d.

End of Definition 15

Here we do not recall the definitions of Dax, Imd, Ifm, Tfm, $\vdash^{fum}$, $\vdash^{mod}$, they can be found in [23].

Instead of "proof method for program verification" we shall simply say "proof method". By a proof method we understand a proof concept $(X \vdash^Y)$ in the sense of Def.11 or one in the sense of Def.9. Thus e.g. $\vdash^F$ and $(Dax \vdash^N)$ are proof methods. When we call $(X \vdash^N)$ a proof method for program verification then what we intuitively have in mind is the proof concept $(X \vdash^N)$ as a device for proving properties of programs. We shall concentrate on the powers of proof methods $(X \vdash^Y)$ to prove partial correctness of programs.

We define a pre-ordering $\leq$ on the proof methods as follows: $(X \vdash^Y) \leq (Y \vdash^Z)$ is defined to hold iff $[(Th \cup X \vdash^Y \rho) \rightarrow (Th \cup Y \vdash^Z \rho)]$ for every similarity type d, $Th \subseteq F_d$ and $\rho \in HF_d$.

The relation $\leq$ induces an equivalence relation $\equiv$ defined as: $(X \vdash^Y) \equiv (Y \vdash^Z)$ iff $[(X \vdash^Y) \leq (Y \vdash^Z)$ and $(Y \vdash^Z) \leq (X \vdash^Y)]$.

A straight line $X \vdash^Y \!\!\!\!\!\!\diagup\; Y \vdash^Z$ on Fig.1 indicates the relation $(X \vdash^Y) \leq (Y \vdash^Z)$. A line with $\neq$ added like $X \vdash^Y \!\!\!\!\!\!\diagup\!\!\!\!\!\!\diagup\; Y \vdash^Z$ indicates the strict relation $<$ that is $[(X \vdash^Y) \not\geq (Y \vdash^Z)$ and $(X \vdash^Y) \leq$ $\leq (Y \vdash^Z)]$. A line with $=?$ added like $X \vdash^Y \!\!\!\!\!\!\diagup\!\!\!\!\!\!\diagup\; Y \vdash^Z$ indicates that $(X \vdash^Y) \leq (Y \vdash^Z)$ but we do not know whether $(X \vdash^Y) \geq (Y \vdash^Z)$ holds or not. Broken line $X \vdash^Y \;\not\leq\; Y \vdash^Z$ with $\not\leq$ indicates that $(X \vdash^Y) \not\leq$ $\not\leq (Y \vdash^Z)$ (but we do not know whether $(X \vdash^Y) \geq (Y \vdash^Z)$ holds or not). If $(X \vdash^Y) \not\leq (Y \vdash^Z)$ is not indicated (either by $\neq$ or by $\not\leq$) then we do not know whether or not $(X \vdash^Y) \leq (Y \vdash^Z)$. Hence "$=?$" is used only to stress that we do not know whether equivalence holds. If two nodes are not connected then we do not know whether they are related in any direction or not that is we do not know whether they are

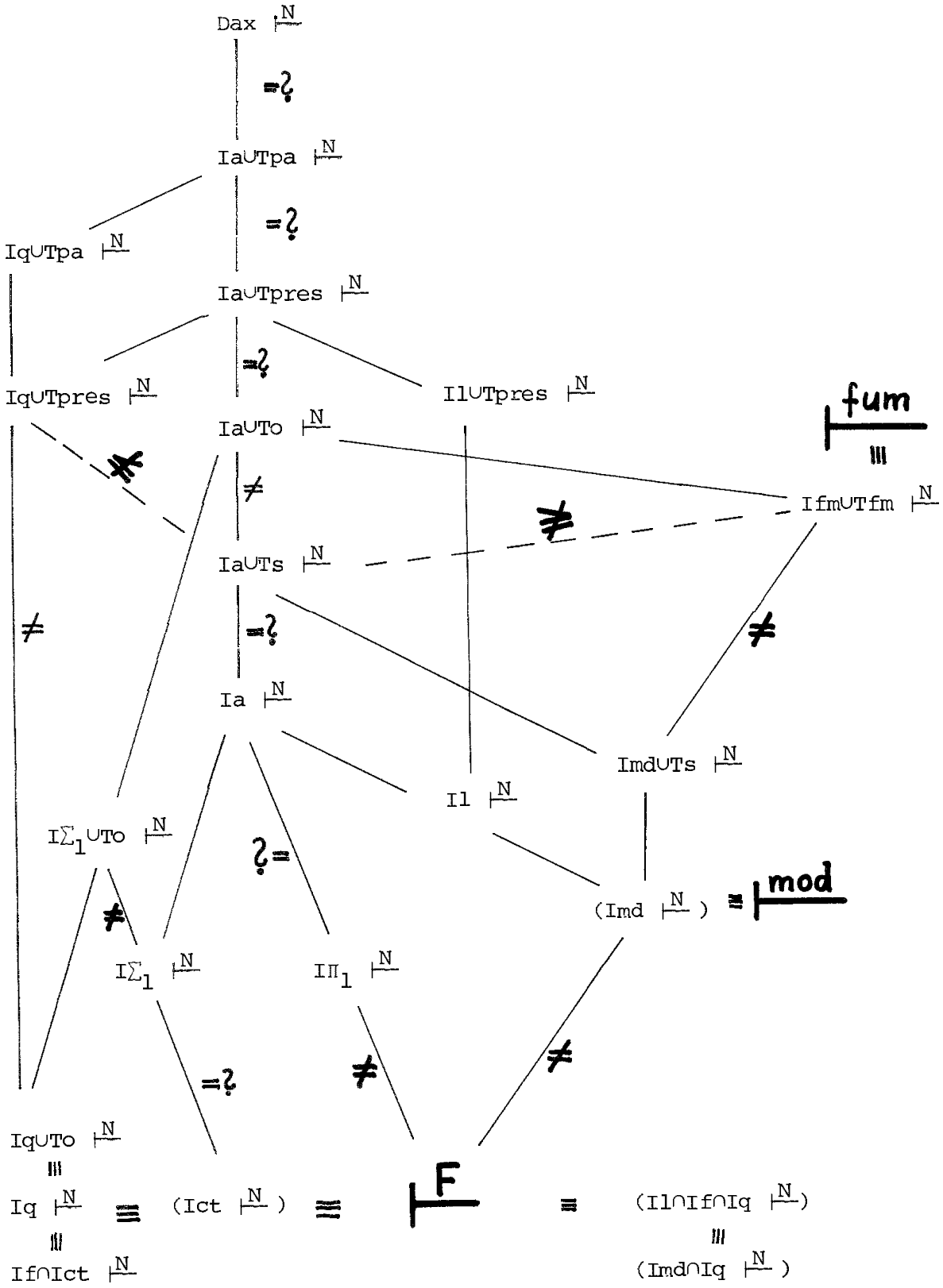
Dax ⊢N
=?
IaUTpa ⊢N
=?
IqUTpa ⊢N
IaUTpres ⊢N
=?
IqUTpres ⊢N
IlUTpres ⊢N
IaUTo ⊢N
fum
≡
IfmUTfm ⊢N
≠
≠
≠
IaUTs ⊢N
=?
≠
Ia ⊢N
ImdUTs ⊢N
Il ⊢N
IΣ1UTo ⊢N
?=
(Imd ⊢N) ≡ mod
≠
IΣ1 ⊢N
IΠ1 ⊢N
=?
≠
≠
IqUTo ⊢N
≡
Iq ⊢N ≡ (Ict ⊢N) ≡ F ≡ (IlIfIq ⊢N)
≡
(ImdIq ⊢N)
If∩Ict ⊢N

FIGURE 1

comparable. For example we do not know whether $(Iq \cup Tpres \vdash^N) \leq$ $\leq (Ia \cup To \vdash^N)$ holds or not. Note that the fact $Iq \not\models Iq \cup To$ does not imply $(Iq \vdash^N) \not\leq (Iq \cup To \vdash^N)$ since proof methods here are compared only w.r.t. $Th \subseteq F_d$ and $\rho \in HF_d$.

In [23], it is indicated for all statements in Fig.1 where to find proofs for them, e.g. the proof of $Ia \cup To \vdash^N) \not\leq (Ia \cup Ts \vdash^N)$ can be found in [23]. In the present paper we shall prove $(I\Pi_1 \vdash^N) \not\leq \vdash^F$, see Theorem 2. Note the contrast between $(Iq \vdash^N) \equiv \vdash^F$ and $(I\Pi_1 \vdash^N) \not\leq \not\leq \vdash^F$.

§4. THE MAIN RESULT OF THE PRESENT PAPER

Theorem 2 below says that more programs can be proved to be correct by using induction over universally quantified formulas than by induction over quantifier-free formulas. (Note that $(Iq \vdash^N) \equiv \vdash^F$.)

THEOREM 2 There are a similarity type d, $p \in P_d$, $\psi \in F_d$ and a finite $Th \subseteq F_d$ such that $(Th \cup I\Pi_1) \models \Box(p,\psi)$ but $Th \not\vdash^F \Box(p,\psi)$.

PROOF. Let d consist of the function symbols $0'$, suc with arities $0,1$ respectively. Let p be the program illustrated on Figure 2. Note that in defining p we use fewer labels than required in the formal definition of P_d, but it is easy to see that this change is not essential while it considerably simplifies the traces of p.

Let $Th \subseteq F_d$ be a finite axiomatization of the theory of $\langle \omega, 0, suc \rangle$ where suc is the usual successor function on ω. Let ψ be the formula $x_3 = x_4$.

We shall prove that $Th \cup I\Pi_1 \models \Box(p,\psi)$ but $Th \not\vdash^F \Box(p,\psi)$.

LEMMA 2.1. $Th \cup I\Pi_1 \models \Box(p,\psi)$.

Proof. Let $\mathfrak{M} = \langle \underset{\sim}{T}, \underset{\sim}{D}, I, ext \rangle \in Mod_{td}(Th \cup I\Pi_1)$ be arbitrary and let $s \in {}^6I$ be a trace of p in $\mathfrak{M}$.

Let $\Sigma_0 F_{td} \overset{d}{=} \{\varphi \in F_{td} : \varphi$ contains no qunatifier$\}$. Then $\Sigma_0 F_{td} \subseteq \subseteq (\Sigma_{0,t} F_{td})$.

Notation: $s_i(z)$ denotes the term $ext(s_i, z)$.

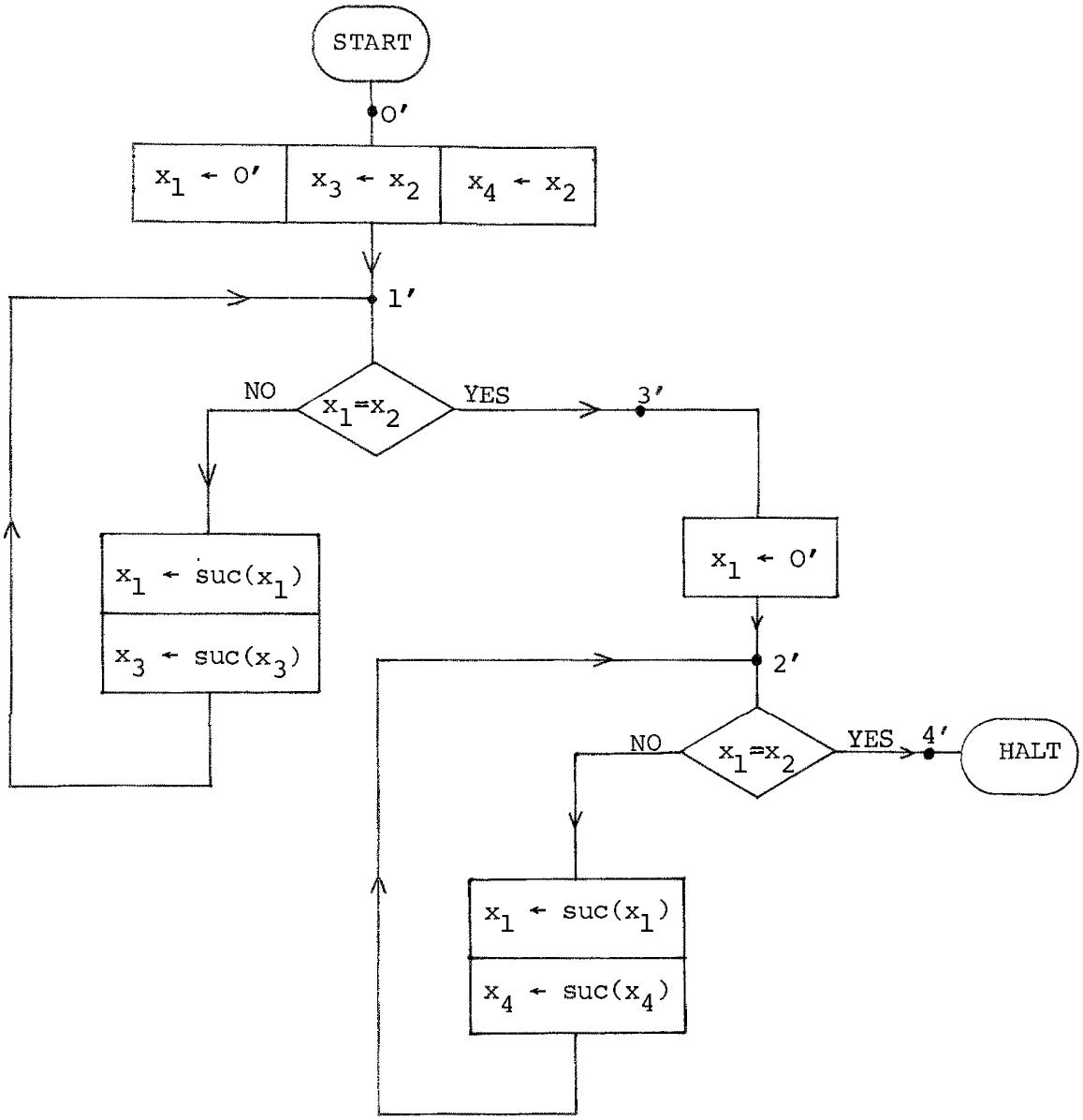

FIGURE 2

We shall extensively use formulas with parameters, e.g. if $\varphi(z_0,y_0)\in F_{td}$ and $s_2\in I$ then we shall sloppily say that $\varphi(z_0,s_2)$ is a formula. Of course, it is not, but the sloppy proof calling $\varphi(z_0,s_2)$ a formula can be translated to a precise one using $\varphi(z_0,y_0)$ and considering the case when y_0 is valuated to denote s_2.

CLAIM 2.1.1. $(\forall z_0)(s_2(z_0)=s_2(0))$.

Proof. Let $\varphi(z_0)$ be the formula $s_2(z_0)=s_2(0)$. Then $\varphi\in\Sigma_0 F_{td}$

hence $\mathrm{ind}(\varphi,z_0)\in Iq$. Clearly $\varphi(0)$. Let $b\in T$ and assume $\varphi(b)$. Since s is a trace, $s_2(sc(b))=s_2(b)=s_2(0)$. Thus $\varphi(sc(b))$. Thus $\mathfrak{M}\models\forall z_0(\varphi(z_0)\rightarrow\varphi(sc(z_0))$. Thus $\mathfrak{M}\models\forall z_0\varphi(z_0)$. QED(2.1.1.)

CLAIM 2.1.2. $\forall z_0[s_5(z_0)\in\{1',3'\}\rightarrow s_4(z_0)=s_2(0)]$.

Proof. Let $\gamma(z_0)$ be the formula $[s_0(z_0)\in\{1',3'\}\rightarrow s_4(z_0)=s_2(0)]$. Clearly $\gamma(0)$, since $s_0(0)=0'\notin\{1',3'\}$ by $\mathfrak{M}\models Th$. Since s is a trace, we have $\forall z_0(\gamma(z_0)\rightarrow\gamma(sc(z_0))$, since if $s_5(z_0)=1'$ then $s_4(sc(z_0))=s_4(z_0)$ and if $s_5(z_0)\in\{3',2',4'\}$ then $s_5(sc(z_0))\notin\{1',3'\}$ hence $\gamma(sc(z_0))$ is obviously true, and if $s_5(z_0)=0'$ then $s_4(sc(z_0))=s_2(z_0)=s_2(0)$ by 2.1.1. Since $\gamma(z_0)\in\Sigma_0F_{td}$, we have $\mathrm{ind}(\gamma,z_0)\in Iq$ and hence by the above, $\mathfrak{M}\models\forall z_0\gamma(z_0)$. QED(2.1.2.)

Let $\chi(z_0,z_1)$ be the formula $[s_5(z_0)=2'\wedge s_5(z_1)=1'\wedge s_1(z_0)=s_1(z_1)]$.

CLAIM 2.1.3. $\forall z_0z_1[\chi(z_0,z_1)\rightarrow s_4(z_0)=s_3(z_1)]$.

Proof. Let $\varkappa(z_0,z_1)$ be the formula $[\chi(z_0,z_1)\rightarrow s_4(z_0)=s_3(z_1)]$. Let $\varphi(z_0)$ be the formula $\forall z_1\varkappa(z_0,z_1)$. Then we have

(1*) $\mathrm{ind}(\varphi(z_0),z_0)\in I\Pi_1$

since $\varkappa\in\Sigma_0F_{td}$. Clearly,

(2*) $\varphi(0)$ holds,

since $s_5(0)\neq2'$ hence $\neg\chi(0,z_1)$. Next we shall prove $\forall z_0(\varphi(z_0)\rightarrow\varphi(sc(z_0))$. To this end, let $b\in T$ and assume that

(3*) $\varphi(b)$.

We want to prove $\varphi(sc(b))$, which amounts to proving $\forall z_1\varkappa(sc(b),z_1)$. Therefore let $\rho(z_1)$ be the formula $\varkappa(sc(b),z_1)$. Then $\rho(z_1)$ is $[\chi(sc(b),z_1)\rightarrow s_4(sc(b))=s_3(z_1)]$. Clearly

(4*) $\rho(0)$

since $\neg\chi(z_0,0)$ for all z_0. To prove $\forall z_1[\rho(z_1)\rightarrow\rho(sc(z_1))]$ let $k\in T$ and assume that

(5*) $\rho(k)$ holds.

We want to prove $\rho(sc(k))$. If $\neg\chi(sc(b),sc(k))$ then $\rho(sc(k))$ obviously holds, assume therefore

(6*) $\chi(sc(b), sc(k))$.

Therefore we have

(7*) $s_5(sc(b))=2'$ and $s_1(sc(b))=s_1(sc(k))$ and $s_5(sc(k))=1'$

by the definition of χ. Since s is a trace of p, these imply

(8*) $s_5(b)\in\{2',3'\}$ and $s_5(k)\in\{0',1'\}$.

CASE 1 $s_5(b)=3'$.
Since s is a trace, then $s_1(sc(b))=0'$. By (7*), $s_1(sc(k))=$ $=s_1(sc(b))=0'$. Then $s_5(k)=0'$, since otherwise $s_1(sc(k))=$ $=suc(s_1(k))\neq 0'$ would be the case by Th and by s being a trace. (Namely, Th $\models \forall x\ suc(x)\neq 0'$.) Then $s_3(sc(k))=s_2(k)=s_2(0)=s_4(b)=$ $=s_4(sc(b))$, by 2.1.1, 2.1.2. Thus $\rho(sc(k))$ is proved for CASE 1.

CASE 2 $s_5(b)=2'$.
Then by (7*), $s_1(sc(b))=suc(s_1(b))$. Thus if $s_5(k)=0'$ were the case then $s_1(sc(k))=0'\neq suc(s_1(b))=s_1(sc(b))$ would follow (by Th and by s being a trace) contradicting (7*). This proves

(9*) $s_5(k)=1'$.

Thence by (7*) we have $s_1(sc(k))=suc(s_1(k))$ since s is a trace. By (7*) then $suc(s_1(k))=s_1(sc(k))=s_1(sc(b))=suc(s_1(b))$. Since Th $\models$ $\models (suc(x_1)=suc(x_2) \rightarrow x_1=x_2)$ and since $\underset{\sim}{D} \models$ Th (i.e. $\mathfrak{M} \models$ Th), this implies $s_1(k)=s_1(b)$. This together with (9*) and with $s_5(b)=2'$ implies $\chi(b,k)$. By (3*) we have $\varphi(b)$ which implies $\varkappa(b,k)$. Therefore $s_4(b)=s_3(k)$. By $\chi(b,k)$ and $\chi(sc(b), sc(k))$, see (6*), and by s being a trace of p, this implies $s_4(sc(b))=suc(s_4(b))=$ $=suc(s_3(k))=s_3(sc(k))$. This proves $\rho(sc(k))$, for CASE 2.

By Cases 1-2 above we proved $\rho(sc(k))$ from the assumption $\rho(k)$, see (5*). By the choice of k we proved $\mathfrak{M} \models \forall z_1[\rho(z_1) \rightarrow$ $\rightarrow \rho(sc(z_1))]$. This together with (4*) implies $\forall z_1 \rho(z_1)$, because by $\rho(z_1)\in\Sigma_0 F_{td}$ we have $ind(\rho(z_1),z_1)\in Iq$. By definition, $\forall z_1\rho(z_1)$ is identical with the formula $\varphi(sc(b))$, hence we proved $\varphi(sc(b))$ from the assumption $\varphi(b)$, see (3*). Thus we proved

(10*) $\forall z_0[\varphi(z_0) \rightarrow \varphi(sc(z_0))]$.

By (1*) we have $ind(\varphi(z_0),z_0)\in I\Pi_1$. By (2*) and (10*) and by $\mathfrak{M} \models I\Pi_1$ we have $\mathfrak{M} \models \forall z_0\varphi(z_0)$. QED(2.1.3.)

Notation: Let $b\in T$. Then $\bar{s}(b) \stackrel{d}{=} \langle s_i(b) : i\in 6\rangle$ and $\bar{\bar{s}}(b) \stackrel{d}{=} \langle s_i(b) : i\in 5\rangle$.

Let $e\in T$ be such that $s_5(e)=4'$. (Note that $4'$ is the label of the HALT-command in p.) We want to prove $s_3(e)=s_4(e)$.

Let $\gamma(z_0)$ be the formula $(\bar{s}(z_0)\neq\bar{s}(e))$. Then $ind(\gamma(z_0),z_0)\in Iq$. Clearly $\gamma(0)$ holds. Assume $\forall z_0(\gamma(z_0)\rightarrow\gamma(sc(z_0))$. Then by $\mathfrak{M}\models Iq$ we have $\forall z_0\gamma(z_0)$, a contradiction, since $\neg\gamma(e)$ is true. Thus we proved $(\exists c\in T)\neg[\gamma(c)\rightarrow\gamma(sc(c))]$. This means $\gamma(c)$ and $\neg\gamma(sc(c))$. Let this c be fixed. We proved $\bar{s}(c)\neq\bar{s}(e)$ and $\bar{s}(sc(c))=\bar{s}(e)$. Since s is a trace and $s_5(sc(c))=s_5(e)=4'$ we have $s_5(c)\neq 4'$ and hence $s_5(c)=2'$. By $s_5(sc(c))=4'$ and since s is a trace then $s_1(c)=s_2(c)=s_2(0)$ by 2.1.1 and $\bar{\bar{s}}(c)=\bar{\bar{s}}(sc(c))=\bar{\bar{s}}(e)$. We have

(11*) $s_5(c)=2'$ and $s_1(c)=s_2(0)$ and $\bar{\bar{s}}(c)=\bar{\bar{s}}(e)$.

Let $\gamma(z_0)$ be the formula $[s_5(z_0)\in\{2',4'\}\wedge s_3(z_0)=s_3(e)]$. Then $ind(\neg\gamma(z_0),z_0)\in Iq$ and $\neg\gamma(0)$ and $\gamma(c)$ imply that $(\exists b\in T)(\neg\gamma(b)\wedge\gamma(sc(b)))$. Since s is a trace, we have

(12*) $\forall z_0(s_5(z_0)\in\{2',4'\}\rightarrow[s_5(sc(z_0))\in\{2',4'\}\wedge s_3(z_0)=s_3(sc(z_0))])$.

By (12*) we have $\forall z_0(s_5(z_0)\in\{2',4'\}\rightarrow[\gamma(z_0)\rightarrow\gamma(sc(z_0))])$. This proves $s_5(b)\notin\{2',4'\}$. Then since s is a trace and $s_5(sc(b))\in\{2',4'\}$ we have $s_5(b)=3$. Since s is a trace, this implies

(13*) $s_5(b)=3$ and $s_3(b)=s_3(sc(b))=s_3(e)$, by $\gamma(sc(b))$.

Let $\gamma(z_0)$ be the formula $[s_5(z_0)=3'\wedge s_3(z_0)=s_3(e)]$. By (13*) we have $\gamma(b)$. Since $ind(\neg\gamma(z_0),z_0)\in Iq$ and $\neg\gamma(0)$, by $\gamma(b)$ we infer that $(\exists a\in T)[\neg\gamma(a)\wedge\gamma(sc(a))]$. Thus $s_5(sc(a))=3'$ and $s_3(sc(a))=s_3(e)$. Since s is a trace this implies the following

(14*) $s_5(a)=1'$ and $s_3(a)=s_3(sc(a))=s_3(e)$ and $s_1(a)=s_2(a)=s_2(0)$,

by 2.1.1. By (11*) and by (14*), $\chi(c,a)$ holds since $s_1(c)=s_2(0)=s_1(a)$. Then 2.1.3 implies $s_4(c)=s_3(a)$. By (11*) we have $s_4(e)=s_4(c)$ and by (14*) we have $s_3(e)=s_3(a)$. We proved $s_4(e)=s_3(e)$.

By the choice of e and s then $\mathfrak{M}\models\Box(p,x_3=x_4)$. By the choice of $\mathfrak{M}$ we proved $Th\cup\Pi_1\models\Box(p,x_3=x_4)$. QED(2.1)

Next we prove Th $\not\vdash^{F} \Box(p,\psi)$. We shall prove more, see Lemma 2.2. Let do be the similarity type d with one binary relation symbol $\leq'$ added. Let $\mathrm{Tho} \subseteq F_{do}$ be the complete theory of $\underset{\sim}{\omega} \overset{d}{=} \langle \omega, 0, \mathrm{suc}, \leq \rangle$, where $\leq$ is the usual ordering in ω. Let $\varphi \in F_{do}$ be any formula such that the variable x_2 does not occur in φ and such that $\mathrm{Tho} \models \models \exists x_1 \exists x_3 \exists x_4 \varphi$.

LEMMA 2.2. $\mathrm{Tho} \not\vdash^{F} (\varphi \to \Box(p,\psi))$.

Proof. Let $\bar{x} = \langle x_1, x_2, x_3, x_4 \rangle$. Suppose $\mathrm{Tho} \vdash^{F} (\varphi \to \Box(p,\psi))$. Then there is a Floyd-Hoare proof $w \in \mathrm{Prf}$ of $(\varphi \to \Box(p,\psi))$ such that w contains inductive assertions attached to the labels 1' and 2' of the program p (see Def.15). Let $\Phi_1(\bar{x})$, $\Phi_2(\bar{x}) \in F_{do}$ be these inductive assertions attached to the labels 1' and 2' of p. We define

$$\begin{aligned}
\mathrm{Fl}(\bar{x}) \overset{d}{=} \{ & \varphi \to \Phi_1(0', x_2, x_2, x_2), \\
& (\Phi_1(\bar{x}) \wedge x_1 \neq x_2) \to \Phi_1(\mathrm{suc}(x_1), x_2, \mathrm{suc}(x_3), x_4), \\
& (\Phi_1(\bar{x}) \wedge x_1 = x_2) \to \Phi_2(0', x_2, x_3, \mathrm{suc}(x_4)), \\
& (\Phi_2(\bar{x}) \wedge x_1 \neq x_2) \to \Phi_2\ \mathrm{suc}(x_1), x_2, x_3, \mathrm{suc}(x_4)), \\
& (\Phi_2(\bar{x}) \wedge x_1 = x_2) \to x_3 = x_4 \}.
\end{aligned}$$

Then our hypothesis $\mathrm{Tho} \vdash^{F} (\varphi \to \Box(p, x_3 = x_4))$ implies $\mathrm{Tho} \vdash \mathrm{Fl}(\bar{x})$. Then $\underset{\sim}{\omega} \models \mathrm{Fl}(\bar{x})$.

Let $R \overset{d}{=} \{ \langle a_1, a_2, a_3, a_4 \rangle \in {}^4\omega : a_1 \leq a_2,\ a_3 = 2a_2,\ a_4 = a_2 + a_1 \}$. Here + is the usual addition of natural numbers and $2a_1 = a_2 + a_2$.

CLAIM 2.2.1. $(\forall \langle a_1, a_2, a_3, a_4 \rangle \in R)\ \underset{\sim}{\omega} \models \Phi_2[a_1, a_2, a_3, a_4]$.

Proof. Let $\langle a_1, a_2, a_3, a_4 \rangle \in R$ be arbitrary. Since $\mathrm{Tho} \models \exists x_1 \exists x_3 \exists x_4 \varphi$, there are $e_1, e_3, e_4 \in \omega$ such that $\underset{\sim}{\omega} \models \varphi[e_1, e_3, e_4]$. Let $\mathfrak{M} \overset{d}{=} \langle \underset{\sim}{\omega}, \underset{\sim}{\omega}, {}^{\omega}\omega, \mathrm{ext} \rangle \in M_{td}$ where $(\forall s \in {}^{\omega}\omega)(\forall b \in \omega)\mathrm{ext}(s,b) = s(b)$. Consider the execution of the program p in $\mathfrak{M}$ with input $\langle e_1, a_2, e_3, e_4 \rangle$. We denote the trace of p of input $\langle e_1, a_2, e_3, e_4 \rangle$ by s. By the definition of a trace (Def.6), it is easy to see that there is a time point $b \in \omega$ such that the value of the control variable x_5 at time point b is the label of Φ_2, that is $\mathrm{ext}(s_5, b) = s_5(b) = 2$, and the values of the program variables at time point b are: $\langle s_1(b), s_2(b), s_3(b), s_4(b) \rangle = \langle a_1, a_2, a_3, a_4 \rangle$. Then, by $\underset{\sim}{\omega} \models \mathrm{Fl}(\bar{x})$ and

$\underset{\sim}{\omega} \models \varphi[s_1(0),s_3(0),s_4(0)]$ and by the definition of a trace we have $\underset{\sim}{\omega} \models \Phi_2[s_1(b),s_2(b),s_3(b),s_4(b)]$. (QED 2.2.1)

By the definition of R we have that $\langle j,2j,4j,3j\rangle \in R$ for every $j\in\omega$. Then by 2.2.1 we have $\underset{\sim}{\omega} \models \Phi_2[j,2j,4j,3j]$ for every $j\in\omega$.

Let U be a nonprincipal ultrafilter over ω. Let $k \overset{d}{=} \overset{d}{=} \langle j : j\in\omega\rangle/U$. For every $n\in\omega$ let $nk \overset{d}{=} \langle nj : j\in\omega\rangle/U$. Let $\underset{\sim}{D}$ be the ultrapower $\underset{\sim}{D} = {}^{\omega}\underset{\sim}{\omega}/U$ of $\underset{\sim}{\omega}$. Then by Los lemma we have $\underset{\sim}{D} \models \models \Phi_2[k,2k,4k,3k]$. See Figure 3!

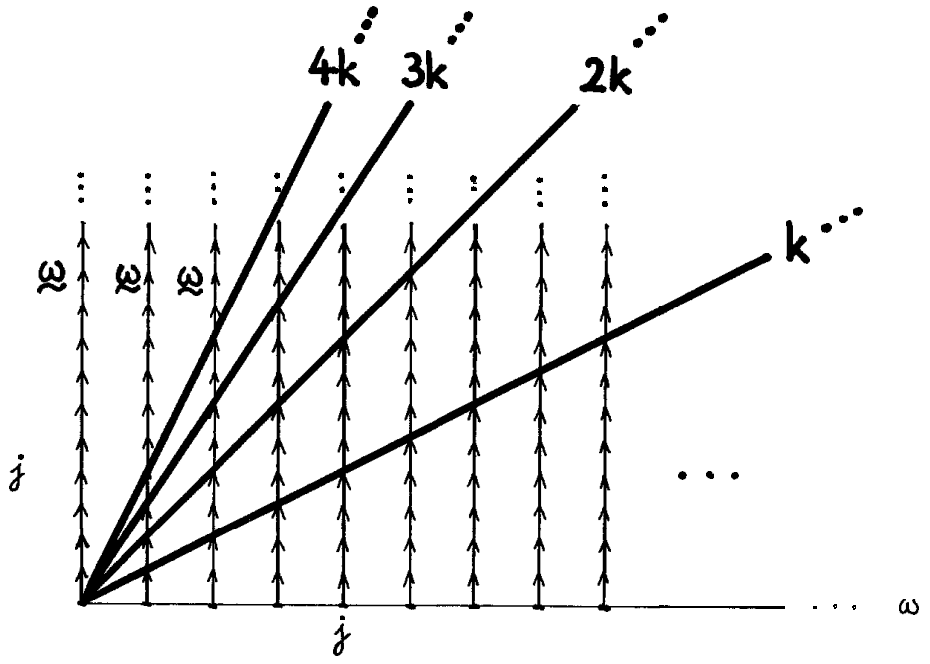

FIGURE 3

We define $suc^n : D \to D$ for every $n\in\omega$ as follows: $(\forall e\in D)[suc^0(e) \overset{d}{=} e$ and $suc^{n+1}(e) \overset{d}{=} suc(suc^n(e))]$.
It is known that $\underset{\sim}{D} = {}^{\omega}\underset{\sim}{\omega}/U$ looks like as it is illustrated on Fig.4 and it is easy to see that the "distance" between any two of k, $2k$, $3k$, $4k$ is infinite, i.e. $(\forall n\in\omega)[suc^n(k)\neq 2k$ and $suc^n(2k)\neq 3k$ etc.$]$.

Now we define $f : D \to D$ as follows. Let $e\in D$ be arbitrary. If there is an $n\in\omega$ such that $suc^n(e)=4k$ or $suc^n(4k)=e$ then $f(e) \overset{d}{=} suc(e)$, otherwise $f(e) \overset{d}{=} e$. Clearly, f is an automorphism of $\underset{\sim}{D}$. Therefore $\underset{\sim}{D} \models \Phi_2[k,2k,suc(4k),3k]$ by $\underset{\sim}{D} \models \Phi_2[k,2k,4k,3k]$. By Los lemma we have that $H \overset{d}{=} \{j\in\omega : \underset{\sim}{\omega} \models \Phi_2[j,2j,suc(4j),3j]\}\in U$. Since U is an ultrafilter, there is a $j\in H$. Let this j be fixed.

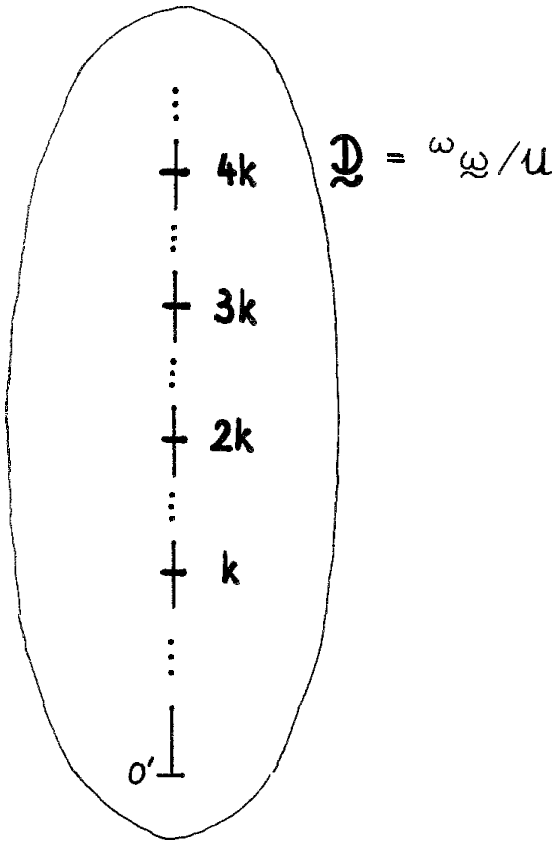

FIGURE 4

Then $\underset{\sim}{\omega} \models \Phi_2[j,2j,suc(4j),3j]$. Since by our hypothesis $\underset{\sim}{\omega} \models Fl(\bar{x})$, we have $\underset{\sim}{\omega} \models ((\Phi_2(\bar{x}) \wedge x_1 \neq x_2) \rightarrow \Phi_2(suc(x_1),x_2,x_3,suc(x_4))$. Thus, from $\underset{\sim}{\omega} \models \Phi_2[j,2j,suc(4j),suc(3j)]$ we can derive $\underset{\sim}{\omega} \models \Phi_2[suc(j),2j,suc(4j),suc(3j)]$ if $j<2j$. If $suc(suc(j))<2j$ then we can repeat this step. Actually, we can repeat this step exactly j times. By induction we can conclude $\underset{\sim}{\omega} \models \Phi_2[2j,2j,suc(4j),4j]$. Then using $\underset{\sim}{\omega} \models ((\Phi_2(\bar{x}) \wedge x_1 = x_2) \rightarrow x_3 = x_4)$ and $\underset{\sim}{\omega} \models Fl(\bar{x})$ we get from $\underset{\sim}{\omega} \models \Phi_2[2j,2j,suc(4j),4j]$ that $\underset{\sim}{\omega} \models (suc(4j)=4j)$, which is a contradiction. Therefore our hypothesis $Tho \overset{F}{\vdash} (\varphi \rightarrow \Box(p,\psi))$ is false.

QED(Theorem 2)

PROBLEM 3 Is $(I\Sigma_1 \overset{N}{\vdash}) \equiv \overset{F}{\vdash}$? I.e. is it true that $(\forall d)(\forall \rho \in HF_d)[I\Sigma_1 \models \rho \Rightarrow \overset{F}{\vdash} \rho]$?

Algebraization of dynamic logic in the spirit of [15] can be found e.g. in [24].

R E F E R E N C E S

[1] Andréka,H. Németi,I., Completeness of Floyd's program verification method w.r.t. nonstandard time models, Seminar Notes Math. Inst. Hungar. Acad. Sci.-SZKI 1977 (in Hungarian). This was abstracted in [2].

[2] Andréka,H. Németi I., Completeness of Floyd Logic, Bull.Section of Logic Wroclaw (1978), Vol 7, No 3, pp.115-121.

[3] Andréka,H. Németi,I. Sain,I., A complete logic for reasoning about programs via nonstandard model theory, Theoret. Comput.Sci., Vol 17(1982) Part I in No 2, pp.193-212, Part II in No 3.

[4] Andréka,H. Németi,I. Sain,I., A complete first order dynamic logic, Preprint, Math.Inst.H.A.S., Budapest, 1980. No.810930.120p.

[5] Andréka,H. Németi,I. Sain,I., Henkin-type semantics for program schemes to turn negative results to positive, Fundamentals of Computation Theory'79 (Proc.Conf.Berlin 1979), Ed. L.Budach, Akademie Verlag, Berlin, 1979. Band 2. pp.18-24.

[6] Andréka,H. Németi,I. Sain,I., A characterization of Floyd-provable programs, In: Mathematical Foundations of Computer Science '81 (Strbské Pleso, Czechoslovakia, 1981), Lecture Notes in Computer Science 118, Springer, Berlin, 1981. pp.162-171.

[7] Burstall,R.M., Program proving as hand simulation with a little induction. IFIP Congress, Stockholm, August 3-10, 1974.

[8] Chang,C.C., Keisler,H.J., Model theory, North-Holland, 1973.

[9] Csirmaz,L., A survey of semantics of Floyd-Hoare derivability, CL&CL - Comput.Linguist, Comput.Lang, 14(1980), pp.21-42.

[10] Csirmaz,L., On the completeness of proving partial correctness, Acta Cybernet., Tom 5, Fasc 2, Szeged, 1981. pp.181-190.

[11] Csirmaz,L. Paris,J.B., A property of 2-sorted Peano models and program verification, Preprint, Budapest, 1981.

[12] Gabbay,D. Pnueli,A. Shelah,S. Stavi,J., On the temporal analysis of fairness, Preprint, Weizmann Inst. of Sci., Dept. of Applied Math., Israel, May 1981.

[13] Gergely,T. Ury,L., Time models for programming logics, In: Mathematical Logic in Computer Science(Proc.Coll. Salgótartján 1978), Colloq.Math.Soc.J.Bolyai Vol 26, North-Holland 1981, pp.359-427.

[14] Hájek,P., Making dynamic logic first-order, In: Mathematical Foundations of Computer Science '81 (Strbské Pleso, Czechoslovakia, 1981) Lecture Notes in Computer Science 118, Springer, Berlin, 1981.

[15] Henkin,L. Monk,J.D. Tarski,A. Andréka,H. Németi I., Cylindric Set Algebras, Lecture Notes in Mathematics 883, Springer-Verlag, Berlin, 1981. v+323p.

[16] Kfoury,D.J. Park,D.M.R., On the termination of program schemes, Information and Control 29(1975), pp.243-251.

[17] Manna,Z. Waldinger,R., Is "Sometime" sometimes better than "Always"? Intermittent assertions in proving program correctness, Preprint, Stanford Research Inst., Menlo Park, June 1976, No. Z173.

[18] Mirkowska,G., PAL - Propositional Algorithmic Logic, In: Logics of Programs, Lecture Notes in Computer Science 125, Springer-Verlag, Berlin, 1981. pp.23-101.

[19] Monk,J.D., Mathematical Logic, Springer Verlag, 1976.

[20] Németi,I., Nonstandard runs of Floyd-provable programs, this volume.

[21] Németi,I., Hilbert style axiomatization of nonstandard dynamic logic, Preprint, Budapest, 1980.

[22] Németi,I., Results on the lattice of dynamic logics, Preprint, Math. Inst. H.A.S., Budapest, 1981.

[23] Németi,I., Nonstandard dynamic logic, In: Proc. Workshop on Logics of Programs (May 1981, New York) Ed.: D.Kozen, Lecture Notes in Computer Science, Springer-Verlag, to appear.

[24] Németi,I., Dynamic algebras of programs, In: Fundamentals of Computation Theory'81 Proc.(Szeged 1981) Lecture Notes in Computer Science 117, Springer-Verlag, Berlin, 1981. pp.281-290.

[25] Parikh,R., Propositional dynamic logics of programs: A survey, In: Logics of Programs, Lecture Notes in Computer Science 125, Springer-Verlag, Berlin, 1981. pp.102-141.

[26] Richter,M.M. Szabo,M.E., Towards a nonstandard analysis of programs, In: Proc. of the Second Victoria Symp. on nonstandard analysis, Victoria, Britisch Columbia, June 1980. Ed. A.Hurd, Lecture Notes in Math., Springer Verlag, 1981.

[27] Sain,I., There are general rules for specifying semantics: Observations on Abstract Model Theory, CL&CL - Comput.Linguist. Comput.Lang., 13(1979), pp.251-282.

[28] Sain,I., First order dynamic logic with decidable proofs and workeable model theory, In: Fundamentals of Computation Theory '81 (Szeged, Hungary 1981), Lecture Notes in Computer Science 117, Springer, Berlin, 1981. pp.334-340.

[29] Stavi,J., Compactness properties of infinitary and abstract languages, In: Logic Colloquium'77, Ed.s A.Macintyre, L.Pacholski, J.Paris,. North-Holland 1978, pp.263-275.