Theoretical Computer Science 17 (1982) 193–212
North-Holland Publishing Company

A COMPLETE LOGIC FOR REASONING ABOUT PROGRAMS VIA NONSTANDARD MODEL THEORY I*

H. ANDRÉKA, I. NÉMETI and I. SAIN
Mathematical Institute of the Hungarian Academy of Sciences, Budapest, H-1053 Hungary

Communicated by M. Nivat
Received August 1979
Revised July 1980

Introduction

In Computer Science and related fields there have been around logical systems in which reasoning about consequences of *actions* is also possible. I.e. in addition to being able to say "All humans are mortal.", "Sokrates is human." etc. we are also allowed to say "After throwing the switch there will be light." or "After touching the hot stove there will be pain.". The new patterns of thought appearing in these logics are of the kind "After doing action p it will be the case that φ." where φ is a formula of classical logic. These patterns of thought are at the very core of human reasoning and hence such logics have appeared not only in Computer Science but also in e.g. Child Psychology, Developmental Psychology, Linguistics, Philosophical Logic. See e.g. Segerberg [31].

Dynamic logic is intended to be the common backbone of all these and related logics, and accordingly its aim is to find that basic structure which is common in all these logics, that basic structure which makes all of them tick. See e.g. Pratt [26], Segerberg [31], Berman [8], Harel [19].

Propositional Dynamic Logic is flourishing in a healthy, convincing and rather attractive way, its model theory is a clear Kripke style one [26, 19] which fits beautifully into the system of model theories of well-understood logics. Propositional Dynamic Logic is developed coherently along the lines of the well-established General Methodology for doing Nonclassical Propositional Logic, see e.g. [31]. Specially the proof concept of Propositional Dynamic Logic is a decidable one and the set of valid formulas is recursively enumerable, see e.g. Pratt [25], Segerberg [30], Parikh [24]. The notion of a proof concept and the property of its being decidable or not can be found in Definition 10 of the present paper (in Section 4). In short, a proof concept is decidable if the *set of all correct proofs* is decidable.

* Part II will appear in the next issue.

(E.g. the proof concept of classical *first order* logic is decidable.) All the published completeness theorems for Propositional Dynamic Logic are based on decidable proof concepts and therefore are completely satisfactory [24, 25, 30], Berman [7].

First order Dynamic Logic (see e.g. [19]) is in a state which is not nearly as satisfactory. There are several different alternative model theories around, these are fairly ad-hoc and the proof concepts used are not decidable, and most of the proposed model theories are such that the set of the valid formulas is not recursively enumerable. We shall refer to the semantics (or model theory) used in [19] as standard (because it refers implicitly to the standard model of arithmetic).

First we tried to select a very small sublanguage S of First order Dynamic Logic such that an acceptable completeness theorem could be proved for S (at least). It turned out in Andréka–Németi–Sain [4] that this is impossible with standard semantics *even if* we are extremally permissive about the choice of S. To alleviate this problem, the standard literature started to use undecidable proof concepts, e.g. infinitely long proofs [19] Section 3.4.2, the so-called Effective ω-rule (which is not decidable either), the so-called Arithmetical Axiomatization of [19] etc. However, there is a nonstandard literature too, (e.g. Csirmaz [14, 15], Andréka–Németi [1–6], Sain [29], Gergely–Úry [18], Biró [9]) which reacted differently to the negative results.

The present paper belongs to this nonstandard school. (This nonstandard school is sometimes called Explicit Time school; see Section 7 of the present paper.) One of our theses is that since there does exist a well-established methodology for doing First order Nonclassical Logic (e.g. Gallin [16], Bowen [10]) by using this methodology one could develop a less ad-hoc model theory for First order Dynamic Logic (similarly to that what is happening with Propositional Dynamic Logic). I.e. one of our aims is to make First order Dynamic Logic fit better into the already existing culture of Nonclassical Logics. Some of our considerations in this line are collected in Section 9.

Another aim of the nonstandard school is to find *decidable proof concepts* for the logics under investigation. In Definition 13 of this paper a proof concept $\vdash^{N}$ is introduced for First order Dynamic Logic which is decidable. In Theorem 2 this proof concept $\vdash^{N}$ is proved to be strongly complete, i.e. for every theory Th and formula φ of First order Dynamic Logic we have Th $\models \varphi$ iff Th $\vdash^{N} \varphi$. The proof concept $\vdash^{N}$ provides also a new proof method for program verification. In Section 6 the new proof method $\vdash^{N}$ is compared with some old ones. It is proved e.g. that $\vdash^{N}$ is strictly stronger than the so-called Floyd–Hoare method.

Concerning the syntax of Dynamic Logic we followed the ideas presented in Pratt [27]. Hence the syntax of Dynamic Logic in the present paper is slightly different from the one in e.g. [25] but this difference does not affect the expressive power of the language. The difference is that our 'action terms' or 'programs' are not structured. The present paper can be translated to the original structured syntax of Dynamic Logic without changing any of the results, see Sain [28].

On the connection with Henkin's higher order model theory

Higher order Logic is strongly incomplete w.r.t. its standard model theory. Henkin devised a nonstandard model theory for Higher order Logic, see [22]. Higher order Logic is complete w.r.t. Henkin's nonstandard model theory [22]. Henkin's nonstandard model theory proved to be rather useful and satisfactory e.g. in getting a deeper understanding of the nature of higher order reasoning, in applications in computer science etc. Gallin [16] adopted Henkin's nonstandard method to Intensional Logic. We are applying Henkin's nonstandard method to First order Dynamic Logic and Logics of Programs. (This will be especially outstanding in Definition 16 and Theorem 7.) By postulating appropriate axioms we can keep our models to be no more nonstandard than Henkin's models are. If Henkin's nonstandard approach was useful for Higher order Logic we do not see why it would not be useful for First order Dynamic Logic too. For more in this line see [29].

Notations

In the following list we shall recall some standard notations, used throughout the paper, from textbooks on logic (mainly from Chang–Keisler [12], Monk [22]). The reader is advised to skip this list and use it only when needed.

d	denotes an arbitrary similarity type of classical one-sorted models. I.e. d correlates arities (natural numbers) to function symbols and relation symbols, see Definition 1(i).
ω	denotes the set of natural numbers such that $0 \in \omega$.
$X = \{x_w : w \in \omega\}$	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. [12, p. 22].
τ	denotes a term of type d in the usual sense of logic, see [12, p. 22] or [22, p. 166, Definition 10.8(ii)].
M_d	denotes the class of all classical one-sorted models of type d, see e.g. [12] or [22, Definition 11.1], or Definitions 1 and 3.

A *classical one-sorted model* is denoted by a bold italic capital like $\boldsymbol{T}$ or $\boldsymbol{D}$ and its *universe* is denoted by the same nonbold capital. E.g. T is the universe of $\boldsymbol{T}$, and D is that of $\boldsymbol{D}$.

By a '*valuation* of the variables' in a model $\boldsymbol{D}$ a function $g: \omega \to D$ is understood, see [22, p. 195].

$\tau[q]_{\boldsymbol{D}}$	denotes the value of the term τ in the model $\boldsymbol{D}$ under the valuation q of the variables, see [12, p. 27, Definition 13.13] or [22, Definition 11.2]. If τ contains no variable, then we write τ instead of $\tau[q]_{\boldsymbol{D}}$, if $\boldsymbol{D}$ is understood.
$\boldsymbol{D} \models \varphi[q]$	denotes that the valuation q satisfies the formula φ in the model $\boldsymbol{D}$.
$L_d = \langle F_d, M_d, \models\rangle$	is the classical first order language of similarity type d, see [29].
td	denotes a certain many-sorted similarity type, see Definitions 3 and 4 and [22].

F_{td} is the set of all classical first order many-sorted formulas of similarity type td, see [22] and Definition 5.

M_{td} is the class of all classical many-sorted models of similarity type td, see [22] and Definitions 3 and 5.

$L_{td} = \langle F_{td}, M_{td}, \models \rangle$ is the classical first order many-sorted language of similarity type td, see [22] and Definition 5.

$\mathfrak{M}$ denotes a classical many-sorted model, usually $\mathfrak{M} \in M_{td}$. The parts of $\mathfrak{M}$ are always denoted as $\mathfrak{M} = \langle T, D, I, \text{ext} \rangle$, see Definitions 3 and 4.

${}^AB$ denotes the set of all functions from A into B, i.e. ${}^AB = \{f : f \text{ maps } A \text{ into } B\}$, see [22, p. 7].

A *function* is considered to be a set of pairs.

Dom f denotes the domain of the function f, and

Rng f denotes the range of the function f, i.e.
Dom $f = \{a : (\exists b)(a, b) \in f\}$,
Rng $f = \{b : (\exists a)(a, b) \in f\}$.

A *sequence* s of length n is a function with Dom $s = n \overset{\text{df}}{=} \{0, 1, \dots, n-1\}$.

$\langle U_s : s \in S \rangle$ denotes the function $\{\langle s, U_s \rangle : s \in S\}$. Moreover for any 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 $(\forall x \in S)\, f(x) \overset{\text{df}}{=} \text{Expr}(x)$.

Natural numbers are used in the von Neumann sense i.e. $n = \{0, 1, \dots, n-1\}$ and especially 0 is the empty set.

1. Syntax (of program schemes)

The following is basically the same as the content of pp. 242–244 in Manna [21]. 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 the set of all constant terms of type d, i.e. d-type terms which do not contain variable symbols.

Logical symbols: $\{\wedge, \neg, \exists, =\}$.

Other symbols: $\{\leftarrow$, **if**, **goto**, **halt**, (,), :$\}$.

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

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

– $(i:$ **if** χ **goto** $v) \in U_d$ if $i, v \in \text{Lab}$, $\chi \in F_d$ is a formula without quantifiers;

– $(i:$ **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 $(i:$ **if** χ **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

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

where $n \in \omega$, $(i_m : u_m) \in U_d$ for $m \leq n$ etc.

An example for a program scheme can be found between Definition 16 and Theorem 7 in Part II.

The set Lab of labels was chosen the above way for technical reasons only. There are many other possible ways for handling labels. The anomalies arising from the present choice of Lab can be avoided by expanding the type d and by fixing some simple axioms, see IA$^+$ in Definition 14 in Part II.

2. Semantics (of program schemes)

By a language with semantics we understand a triple $L = \langle F, M, \models\rangle$ of classes such that $\models \subseteq M \times F \times \text{Sets}$ 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, and $\models$ the satisfaction relation 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 [23, 29], and Stavi [32, p. 265].

Here we try to develop a natural semantic framework for programs and statements about programs. In trying to understand the 'Programming Situation', its languages, their meanings etc., the first question is how an interpretation or model of a program or program scheme $p \in P_d$ should look like. The classical approach (Manna [21], Ianov [20]) says that an interpretation or model of a program scheme is a relational structure $\boldsymbol{D} \in M_d$ consisting of all the possible *data values*. The program p contains variables, say 'x'. The classical approach says that x denotes elements of D just as variables in classical first order logic do. Now we argue that x does *not* denote elements of D but rather x denotes some kind of 'locations' or 'addresses' which may *contain* different data values (i.e. elements of D) at different points of time. Thus there is a set I of locations, a set T of time points, and a function $\text{ext}: I \times T \to D$ which tells for every location $s \in I$ and time point $b \in T$ what the content of location s is at time point b. Of course, this content $\text{ext}(s, b)$ is a data value, i.e. it is an element of D. Time has a structure too ('later than' etc.) and data values have structure too, thus we have structures $\boldsymbol{T}$ and $\boldsymbol{D}$ over the sets T and D of time points and possible data values respectively. Therefore we shall define a model or interpretation for programs $p \in P_d$ to be a four-tuple $\mathfrak{M} = \langle \boldsymbol{T}, \boldsymbol{D}, I, \text{ext}\rangle$ where $\boldsymbol{T}$ and $\boldsymbol{D}$ are the time structure and data structure resp., I is the set of locations and $\text{ext}: I \times T \to D$ the 'content of ... at time ...' function (see Definition 4). We shall call the elements of I *intensions* instead of locations. The reasons for this and for the name 'ext' are explained in Section 9. For a detailed account of the above considerations see also Section 8.

Of course, when specifying semantics of a programming language P_d we may have ideas about how an interpretation $\mathfrak{M}$ of P_d may look like and how it may not. These ideas may be expressed in the form of *axioms* about $\mathfrak{M}$. E.g. we may postulate that $\boldsymbol{T}$ of $\mathfrak{M}$ has to satisfy the Peano Axioms of arithmetic. For such axioms see Definitions 14, 16 and 19 in Part II. These axioms are easy to express since a closer investigation of $\mathfrak{M}$ defined above reveals that it is a model of classical 3-sorted logic (the sorts being 'time', 'data' and 'intensions'). Thus the axioms can be formed in classical 3-sorted logic (Definition 5) in a convenient manner to express all our ideas or postulates about the semantics of the programming language P_d under consideration.

Now we turn to work out these ideas in detail.

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 $d_1 : \Sigma \to \omega$ for some set Σ, $H \subseteq \Sigma$ and $(\forall r \in \Sigma)\ d_1(r) > 0$.

The elements of Σ are called the *symbols* of d, and the elements of H the *operation symbols* or function symbols of d.

Let $r \in \Sigma$. Then we shall write $d(r)$ instead of $d_1(r)$.

(ii) Let $d = \langle H, d_1\rangle$ be a similarity type, let $\Sigma = \text{Dom}\ d_1$ as above.

By a *model of type d* we understand a pair $\boldsymbol{D} = \langle D, R\rangle$ such that R is a function with $\text{Dom}\ R = \Sigma$ and $(\forall r \in \Sigma)\ R(r) \subseteq {}^{d(r)}D$ and if $r \in H$, then $R(r) : {}^{(d(r)-1)}D \to D$.

Notation. $\langle D, R_r\rangle_{r\in\Sigma} \overset{\text{df}}{=} \langle D, \langle R_r : r \in \Sigma\rangle\rangle \overset{\text{df}}{=} \langle D, R\rangle$.

I.e. $\boldsymbol{D} = \langle D, R_r\rangle_{r\in\Sigma}$ 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 $r \in \Sigma$.

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 $\boldsymbol{D}$.

The set D is called the *universe* of $\boldsymbol{D}$.

(iii) $M_d \overset{\text{df}}{=} \{\boldsymbol{D} : \boldsymbol{D}$ is a model of type $d\}$. □

Definition 2 (the similarity type t of arithmetic and its standard model $\boldsymbol{N}$). t denotes the similarity type of Peano's Arithmetic. In more detail, $t = \langle\{+, \cdot, 0, 1\}, t_1\rangle$ where $\text{Dom}\ t_1 = \{\leq, +, \cdot, 0, 1\}$, $t(\leq) = 2$, $t(+) = t(\cdot) = 3$, and $t(0) = t(1) = 1$.

The standard model $\boldsymbol{N}$ of t will be sloppily denoted as $\langle \omega, \leq, +, \cdot, 0, 1\rangle = \boldsymbol{N}$ instead of the more precise notation $\boldsymbol{N} = \langle \omega, R\rangle$ where $R(\leq) = \{\langle n, m\rangle \in {}^2\omega : n \leq m\}, \ldots, R(1) = 1$.

Note that $\boldsymbol{N} \in M_t$. □

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

Definition 3 (many-sorted models, [22]). Let S be a set. Then S^* denotes the set of all finite strings of elements of S, i.e. $S^* \overset{\text{df}}{=} \bigcup\{{}^nS : n \in \omega\}$.

(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 $m_2 : \Sigma \to S^*$ for some set Σ, $H \subseteq \Sigma$ and $(\forall r \in \Sigma)\ m_2(r) \notin {}^0S$.

The elements of S are called the *sorts* of m.

If $r \in \Sigma$, then we shall write $m(r)$ instead of $m_2(r)$.

(ii) Let m be a many-sorted similarity type and let $\Sigma = \text{Dom}\, m_2$ as above.

By a (many-sorted) model of type m we understand a pair $\langle\langle U_s : s \in S\rangle, R\rangle$ such that R is a function with $\text{Dom}\, R = \Sigma$ and if $r \in \Sigma$ and $m(r) = \langle s_1, \ldots, s_n\rangle$, then $R(r) \subseteq U_{s_1} \times \cdots \times U_{s_n}$ and if in addition $r \in H$, then $R(r)$ is a function $R(r)\colon U_{s_1} \times \cdots \times U_{s_{n-1}} \to U_{s_n}$.

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

$M_m \overset{\text{df}}{=} \{\mathfrak{M} : \mathfrak{M}$ is a many-sorted model of type $m\}$. □

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 Definition 2 and by our convention $\text{Dom}\, d_1 \cap \text{Dom}\, t_1 = \emptyset$.

Now we define td to be td $\overset{\text{df}}{=} \langle S, K, \text{td}_2\rangle$ where

(a) $S \overset{\text{df}}{=} \{t, d, i\}$, $|S| = 3$. (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) $K \overset{\text{df}}{=} \{+, \cdot, 0, 1, \text{ext}\} \cup H$. ($K$ is the set of operation symbols of td.)

(c) $\text{td}_2\colon (\text{Dom}\, t_1 \cup \text{Dom}\, d_1 \cup \{\text{ext}\}) \to S^*$ such that $\text{td}_2(\text{ext}) = \langle i, t, d\rangle$, $\text{td}_2(r) \in {}^n\{t\}$ if $t(r) = n$ and $\text{td}_2(r) \in {}^n\{d\}$ if $d(r) = n$. E.g. $\text{td}_2(\leqslant) = \langle t, t\rangle$, $\text{td}_2(+) = \langle t, t, t\rangle, \ldots, \text{td}_2(1) = \langle t\rangle$.

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

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

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

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

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

Notation.

$$\langle\langle U_t, R_r\rangle_{r\in\text{Dom}\, t_1}, \langle U_d, R_r\rangle_{r\in\text{Dom}\, d_1}, U_i, R_{\text{ext}}\rangle \overset{\text{df}}{=} \langle\langle U_t, U_d, U_i\rangle, R_r\rangle_{r\in\Sigma} = \mathfrak{M}.$$

We define

$$\boldsymbol{T} \overset{\text{df}}{=} \langle U_t, R_r\rangle_{r\in\text{Dom}\, t_1}, \qquad T \overset{\text{df}}{=} U_t,$$

$$\boldsymbol{D} \overset{\text{df}}{=} \langle U_d, R_r\rangle_{r\in\text{Dom}\, d_1}, \qquad D \overset{\text{df}}{=} U_d \quad \text{and} \quad I \overset{\text{df}}{=} U_i.$$

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

$$\langle \boldsymbol{T}, \boldsymbol{D}, I, \text{ext}\rangle \overset{\text{df}}{=} \langle\langle U_t^{\mathfrak{M}}, U_d^{\mathfrak{M}}, U_i^{\mathfrak{M}}\rangle, r^{\mathfrak{M}}\rangle_{r\in\Sigma} \overset{\text{df}}{=} \mathfrak{M}.$$

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

Note that $\mathfrak{M} \in M_{\text{td}}$ iff ($\boldsymbol{T} \in M_t$, $\boldsymbol{D} \in M_d$, and $\text{ext}: I \times T \to D$).

For a more detailed introduction to many-sorted languages, like $L_{\text{td}} = \langle F_{\text{td}}, M_{\text{td}}, \models \rangle$ defined below, the reader is referred e.g. to [22]. If understanding Definitions 3–6 here is hard for the reader, then consulting [22] should help since L_{td} is the most usual classical many-sorted language of similarity type td.

Definition 5 (the first order 3-sorted language $L_{\text{td}} = \langle F_{\text{td}}, M_{\text{td}}, \models \rangle$ of type td, [22]). Let $d = \langle H, d_1 \rangle$ be any one-sorted similarity type. Recall from Definitions 2 and 4 that t is a fixed similarity type, and td 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 $X \overset{\text{df}}{=} \{x_w: w \in \omega\}$, $Y \overset{\text{df}}{=} \{y_w: w \in \omega\}$ and $Z \overset{\text{df}}{=} \{z_w: w \in \omega\}$ 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 the set of all first order formulas of type d with variables in X, and Tm_t^Z the set of all first order terms of type t with variables in Z.

The set $\text{Tm}_{\text{td},d}$ of terms of type td and of sort d is defined to be the smallest set satisfying conditions (1)–(3) below:

(1) $X \subseteq \text{Tm}_{\text{td},d}$,

(2) $\text{ext}(y_w, \tau) \in \text{Tm}_{\text{td},d}$ for any $\tau \in \text{Tm}_t^Z$ and $w \in \omega$,

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

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

(4) $(\tau_1 = \tau_2) \in F_{\text{td}}$ for any $\tau_1, \tau_2 \in \text{Tm}_{\text{td},d}$,

(5) $r(\tau_1, \ldots, \tau_n) \in F_{\text{td}}$ for any $\tau_1, \ldots, \tau_n \in \text{Tm}_{\text{td},d}$ and for any $r \notin H$ if $d(r) = n$,

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

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

(8) if $\varphi, \psi \in F_{\text{td}}$, then

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

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

(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 [22]) which is completely analogous with the one-sorted case. E.g.

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

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

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

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

The formula $\varphi \in F_{td}$ is *valid* 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. □

Definition 6 (the similarity type d' and the standard model $\mathfrak{N}$ of type td′). Let the similarity type d' be a disjoint copy of t, i.e. let $d' = \langle\{+', \cdot', 0', 1'\}, d_1'\rangle$ where $\mathrm{Dom}\, d_1' = \{\leq', +', \cdot', 0', 1'\}$ and $d'(+') = d'(\cdot') = 3$, $d'(0') = d'(1') = 1$ and $d'(\leq') = 2$.

In defining $\mathfrak{N} = M_{td}$, we shall use the Convention at the end of Definition 4. The standard model $\mathfrak{N}$ of type td′ is defined to be $\mathfrak{N} \overset{\text{df}}{=} \langle \boldsymbol{N}, \boldsymbol{N}', {}^{\omega}\omega, \text{ext}\rangle$ where $\boldsymbol{N}$ is the standard model of type t (see Definition 2), $\boldsymbol{N}'$ the same standard model but of type d', and

$$(\forall s \in {}^{\omega}\omega)(\forall b \in N)\ \text{ext}(s, b) = s(b). \quad \square$$

In this paper we shall define several sets of axioms in the language L_{td}, see Definitions 14, 15, 16, 19 in Part II. Each of them will be valid in the standard model $\mathfrak{N}$.

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

Terminology. What we call here a d-type model $\boldsymbol{D} \in M_d$ was called an '*interpretation*' $\mathcal{I}$ in [21, Section 4.1.2]. Definition 7 is a formalized version of the one given in [21, pp. 244–245].

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

$$p = \langle (i_0\colon u_0), \ldots, (i_{n-1}\colon u_{n-1}), (i_n\colon \textbf{halt})\rangle.$$

Further, throughout $c \in \omega$ denotes the least element w of ω for which $(\forall v \geq w)[x_w$ does not occur in $p]$. I.e. $\{x_w\colon w < c\}$ 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.

Notation. Let $\langle \boldsymbol{T}, \boldsymbol{D}, I, \text{ext}\rangle \in M_{td}$, see the Convention at the end of Definition 4. Let $s_0, \ldots, s_m \in I$, $\bar{s} = \langle s_0, \ldots, s_m\rangle$. Let $b \in T$.

Then we define

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

Definition 7 (traces of programs in time-models). Let $p \in P_d$ and $\mathfrak{M} \in M_{td}$. We shall use the Convention in Definition 4 and Convention 1. 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 trace of p in* $\mathfrak{M}$ if the following (i) and (ii) are satisfied:

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

(ii) For every $b \in T$ and for every $j \leqslant c$ if $\mathrm{ext}(s_c, b) = i_m$, then statements (1)–(3) below hold:

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

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

(2) If $u_m =$ "**if** χ **goto** v", then

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

(3) If $u_m =$ "**halt**", then

$\mathrm{ext}(s_j, b+1) = \mathrm{ext}(s_j, b)$. □

Definition 8. Let $s = \langle s_0, \ldots, s_c \rangle$ be a trace of $p \in P_d$ in $\mathfrak{M} \in M_{\mathrm{td}}$.

(i) Let $k \in {}^{\omega}\boldsymbol{D}$. The trace s is said to be *of input* k iff

$$(\forall j < c)\ k(j) = \mathrm{ext}(s_j, 0).$$

I.e. s is of input k iff

$$\langle k_0, \ldots, k_{c-1} \rangle = \langle \mathrm{ext}(s_0, 0), \ldots, \mathrm{ext}(s_{c-1}, 0) \rangle.$$

(ii) Recall from Convention 1 that i_n is the last label of the program p (i.e. i_n is the label of the **halt**-command). Let $b \in T$.

We say that *s terminates p* at time b in $\mathfrak{M}$ iff

$$\mathrm{ext}(s_c, b) = i_n.$$

We shall sometimes write "*s terminates at b*" instead of "s terminates p at time b in $\mathfrak{M}$".

(iii) Let $k, q \in {}^{\omega}\boldsymbol{D}$. We define q to be a *possible output* of p with input 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

$$\langle q_0, \ldots, q_{c-1} \rangle = \langle \mathrm{ext}(s_0, b), \ldots, \mathrm{ext}(s_{c-1}, b) \rangle.$$

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

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 $\langle k_0, \ldots, k_{c-1} \rangle$. □

Remark. A trace $\langle s_0, \ldots, s_c \rangle$ of a program $p \in P_d$ correlates to each variable x_w ($w \leqslant c$) occurring in the program p an intension or '*history*' s_w such that the value

$\text{ext}(s_w, b)$ 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 $\text{ext}(s_w, -): T \to D$ from time points T to data values D. This function is the 'history' of the variable x_w during an *execution of the program p in the model* $\mathfrak{M}$. Definition 7 ensures that the sequence $\langle \text{ext}(s_0, -), \ldots, \text{ext}(s_c, -) \rangle$ 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'.

Observe that a trace is nothing but a valuation of some variables of sort i.
For a valuation s of the variables of sort i into the universe I of $\mathfrak{M}$ we define

$$\mathfrak{M} \models p[s] \text{ iff } \langle s_0, \ldots, s_c \rangle \text{ is a trace of } p \text{ in } \mathfrak{M}.$$

By now we have defined a semantics of program schemes, i.e. we have a language

$$\langle P_d, M_{\text{td}}, \models \rangle.$$

For any set $\text{Th} \subseteq F_{\text{td}}$ of axioms we define $\text{Mod(Th)} \subseteq M_{\text{td}}$ to be the class of all models of Th. Now for every set $\text{Th} \subseteq F_{\text{td}}$ we have a language

$$\text{PL}_{\text{Th}} = \langle P_d, \text{Mod(Th)}, \models \rangle$$

where $\mathfrak{M} \models p[s]$ is defined in Definition 7 for every $\mathfrak{M} \in \text{Mod(Th)}$. We shall call such a language a *programming language with semantics*. But it is not yet a language for reasoning about programs. That comes in Section 3.

To consider *axiomatizable classes* Mod(Th) of interpretations (instead of a single interpretation $\mathfrak{M}$ or all possible ones M_{td}) to be the semantics of P_d, was suggested in works of Burstall and Darlington [11], Courcelle and Guessarian [13], Gergely and Szöts [17, p. 49] etc. We believe that this is a very important point which should be emphasized. We believe that neither considering a single model $\mathfrak{M}$ to be the semantics nor considering M_{td} or M_d to be the semantics could give us really relevant information about semantics of programming. I.e. neither $\langle P_d, \mathfrak{M}, \models \rangle$ nor $\langle P_d, \mathfrak{N}, \models \rangle$ nor $\langle P_d, M_{\text{td}}, \models \rangle$ nor $\langle P_d, M_d$, some meaning function$\rangle$ could give us really deep insights if we would choose them as the central subject of our study. Note that, e.g. in [21], only these two extremes were considered. On the other hand, investigating $\langle P_d, \text{Mod(Th)}, \models \rangle$ for all possible choices of $\text{Th} \subseteq F_{\text{td}}$ *can* give insights, especially when Th is required to be recursively enumerable. Arguments explaining why this is true were given in [13, 17, 23], and in other works on classes of interpretations by Courcelle, Guessarian and their colleagues.

About using Th

It might look counter-intuitive to execute programs in arbitrary elements of M_{td}. However, we can collect *all our postulates about time* into a set $\text{Ax} \subseteq F_{\text{td}}$ of axioms which this way would define the class $\text{Mod(Ax)} \subseteq M_{\text{td}}$ of all *intended interpretations* of P_d. Then we can use the language PL_{Ax}. Such a set Ax of axioms will be proposed in Definition 14 in Part II. *If* one wants to define semantics with unusual time structure e.g. parallelism, nondeterminism, interactions etc. then one can choose an Ax *different* from the one proposed in this paper.

3. Statements about programs

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

Definition 9 (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) $F_{\mathrm{td}} \subseteq \mathrm{DF}_d$.

(2) $(\forall p \in P_d)(\forall \psi \in \mathrm{DF}_d)\,\Box(p, \psi) \in \mathrm{DF}_d$.

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

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 $\mathfrak{M} \in M_{\mathrm{td}}$. Let $\mathfrak{M} = \langle \boldsymbol{T}, \boldsymbol{D}, I, \mathrm{ext}\rangle \in M_{\mathrm{td}}$. 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 $\mathfrak{M} \models \varphi[v]$ for all $\varphi \in \mathrm{DF}_d$.

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

(5) Let $p \in P_d$ and $\psi \in \mathrm{DF}_d$ 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

$$\mathfrak{M} \models \Box(p, \psi)[g, k, r] \text{ iff}$$

$$(\mathfrak{M} \models \psi[g, q, r] \text{ for every possible output } q \text{ of } p \text{ with input } k \text{ in } \mathfrak{M}.)$$

For 'possible output' see Definition 8.

(6) Let $\varphi, \psi \in \mathrm{DF}_d$ 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

$$\mathrm{DL}_d \overset{\mathrm{df}}{=} \langle \mathrm{DF}_d, M_{\mathrm{td}}, \models\rangle$$

where $\models$ is defined in (ii) above. $\Box$

Notation. Let $p \in P_d$ and $\psi \in \mathrm{DF}_d$. 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 following 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 as shown in Table 1.

Table 1

1. p is partially correct w.r.t. output condition ψ	$\Box(p, \psi)$
2. p is partially correct w.r.t. input condition φ and output condition ψ (in Hoare's notation $\varphi\{p\}\psi$)	$(\varphi \rightarrow \Box(p, \psi))$
3. p is totally correct w.r.t. output condition ψ in the weaker sense	$\Diamond(p, \psi)$
4. p is totally correct in the stronger sense	$\Diamond(p, \psi) \wedge \Box(p, \psi)$
5. p terminates	$\Diamond(p, \textbf{true})$
6. if the input of p_2 is the output of p_1, then p_2 is totally correct w.r.t. ψ	$\Box(p_1, \Diamond(p_2, \psi))$
7. p is deterministic	$(\forall x_0 \cdots \forall x_{2c-1}) \left[\Diamond\left(p, \left(\bigwedge_{i<c} x_i = x_{c+i}\right)\right) \rightarrow \Box\left(p, \left(\bigwedge_{i<c} x_i = x_{c+i}\right)\right)\right]$
8. for any fixed input, if some output of p satisfies ψ, then every output of p satisfies ψ	$\Diamond(p, \psi) \rightarrow \Box(p, \psi)$
9. p satisfies the input–output relation $\psi(x_0, \ldots, x_{c-1}, x_c, \ldots, x_{2c-1})$ where $x_c, \ldots, x_{2c-1}$ is the input and $x_0, \ldots, x_{c-1}$ the output	$(\forall x_0 \cdots \forall x_{2c-1}) \left[\left(\bigwedge_{i<c} x_i = x_{i+c}\right) \rightarrow \Box(p, \psi(x_0, \ldots, x_{2c-1}))\right]$
10. p_1 simulates p_2	$(\forall x_0 \cdots \forall x_{2c-1}) \left[\Diamond\left(p_2, \bigwedge_{i<c} x_i = x_{i+c}\right) \rightarrow \Diamond\left(p_1, \bigwedge_{i<c} x_i = x_{c+i}\right)\right]$

We shall use the model theoretic consequence relation in the usual sense. I.e. let $\text{Th} \subseteq DF_d$, $\varphi \in DF_d$ and $K \subseteq M_{td}$. Then

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

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

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

$$\text{Mod(Th)} \overset{\text{df}}{=} \{\mathfrak{M} \in M_{td} : \mathfrak{M} \models \text{Th}\},$$

and

$$\text{Th} \models \varphi \text{ iff } \text{Mod(Th)} \models \varphi.$$

4. Completeness theorem of DL_d

Theorem 1. *Let* $\mathrm{Th} \subseteq \mathrm{DF}_d$ *be recursively enumerable. Then* $\{\varphi \in \mathrm{DF}_d : \mathrm{Th} \models \varphi\}$ *is recursively enumerable.*

Proof. This Theorem 1 is a consequence of Theorem 2 below. See Fact 1 below and Lemmas 3 and 4 in the proof of Theorem 2. □

Notations. Let X be a set. Then X^* denotes the set of all finite sequences of elements of X, i.e. $X^* = \bigcup\{{}^mX : m \in \omega\}$. 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". $\mathrm{Sb}\, X$ denotes the set of all subsets of X.

Definition 10 (proof concept, [22]). Let $L = \langle F, M, \models\rangle$ be a language.

By a *proof concept* on the set F we understand a relation $\vdash \subseteq \mathrm{Sb}(F) \times F$ together with a set $\mathrm{Pr} \subseteq F^*$ such that $\mathrm{Th} \vdash \varphi$ iff $\langle H, w, \varphi\rangle \in \mathrm{Pr}$ for some finite $H \subseteq \mathrm{Th}$ and for some $w \in F^*$.

The proof concept $\langle \vdash, \mathrm{Pr}\rangle$ 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*. □

Sometimes we shall sloppily write "$\vdash$ is a decidable proof concept" instead of "$\langle \vdash, \mathrm{Pr}\rangle$ 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.

Fact 1. Let $\langle \vdash, \mathrm{Pr}\rangle$ be a decidable proof concept on the set F. Let $\mathrm{Th} \subseteq F$ be recursively enumerable. Then $\{\varphi \in F : \mathrm{Th} \vdash \varphi\}$ is recursively enumerable too.

Proof. Well known from the theory of algorithms. □

Theorem 2 (strong completeness of DL_d). (i) *There is a decidable proof concept* $\vdash^{\mathrm{N}}$ *for the language* DL_d *such that for every* $\mathrm{Th} \subseteq \mathrm{DF}_d$ *and* $\varphi \in \mathrm{DF}_d$ *we have*

$$\mathrm{Th} \models \varphi \textit{ iff } \mathrm{Th} \vdash^{\mathrm{N}} \varphi.$$

(ii) *The language* DL_d *is compact.*

Proof. The idea of the proof is to reduce (or translate) the language DL_d to the complete and compact (classical 3-sorted) language $L_{\mathrm{td}} = \langle F_{\mathrm{td}}, M_{\mathrm{td}}, \models\rangle$ by a total computable function $\theta : \mathrm{DF}_d \to F_{\mathrm{td}}$ such that condition (*) below holds.

for every $\varphi \in \mathrm{DF}_d$ and for every $\mathfrak{M} \in M_{\mathrm{td}}$ we have
$\mathfrak{M} \models \varphi$ iff $\mathfrak{M} \models \theta(\varphi)$. (*)

Suppose we have a 'translation' function $\theta: \mathrm{DF}_d \to F_{\mathrm{td}}$ which satisfies (*). Then the proof of Theorem 2 will be the following:

Let $\mathrm{Th} \subseteq \mathrm{DF}_d$, $\varphi \in \mathrm{DF}_d$. By an $\vdash^{\mathrm{N}}$-proof of φ from Th we shall understand a sequence $\langle H, \langle \theta[H], w, \theta(\varphi)\rangle, \varphi\rangle$ such that $H \subseteq \mathrm{Th}$, $|H| < \omega$, and $\langle \theta[H], w, \theta(\varphi)\rangle$ is a classical first order proof of $\theta(\varphi)$ from $\theta[H]$, where $\theta[H] \overset{\mathrm{df}}{=} \{\theta(\psi): \psi \in H\}$.

Since $\theta: \mathrm{DF}_d \to F_{\mathrm{td}}$ is a total computable function, our proof concept $\vdash^{\mathrm{N}}$ will be decidable. Then using Gödel's completeness theorem for L_{td} and property (*) of θ we shall prove the completeness theorem for DL_d.

First we define the function $\theta: \mathrm{DF}_d \to F_{\mathrm{td}}$.

Convention 2. Instead of z_0 we shall often write z.

Notation. Let $\chi \in F_d$ be a formula without quantifiers. Let $\bar{x} = \langle x_0, \dots, x_e\rangle$ be a sequence of variables from X such that the variables occurring in χ are among $\{x_0, \dots, x_e\}$. Then $\chi[\mathrm{ext}(\bar{y}, z)]$ denotes the formula obtained from χ such that x_j is substituted in χ everywhere by $\mathrm{ext}(y_j, z)$ for each $j \leq e$.

If τ is a term of type d and with variables in $\{x_0, \dots, x_e\}$, then we use the notation $\tau[\mathrm{ext}(\bar{y}, z)]$ similarly.

Note that $\chi[\mathrm{ext}(\bar{y}, z)] \in F_{\mathrm{td}}$ and $\tau[\mathrm{ext}(\bar{y}, z)] \in \mathrm{Tm}_{\mathrm{td},d}$.

Definition 11 (the function $\theta: \mathrm{DF}_d \to F_{\mathrm{td}}$). The definition of θ goes in 2 steps.

(1) First we define a function $\mu: P_d \times \omega \to F_{\mathrm{td}}$.

Let $p = \langle (i_0: u_0), \dots, (i_n: \textbf{halt})\rangle \in P_d$ be fixed. Recall from Convention 1 that c is associated to p such that the variables occurring in p are among $x_0, \dots, x_{c-1}$, and x_c is the control variable of p.

Let $\bar{y} = \langle y_0, \dots, y_c\rangle$. First we define auxiliary formulas $\nu(p)$, $\nu_m(p, z, \bar{y})$ for every $m \leq n$ by statements (i)–(iv) below:

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

$$\nu_m(p, z, \bar{y}) \overset{\mathrm{df}}{=} \Big[\mathrm{ext}(y_c, z+1) = i_{m+1} \wedge \mathrm{ext}(y_w, z+1) = \tau[\mathrm{ext}(\bar{y}, z)] \\ \wedge \bigwedge_{\substack{j<c \\ j \neq w}} \mathrm{ext}(y_j, z+1) = \mathrm{ext}(y_j, z)\Big].$$

(ii) If $u_m =$ "**if** χ **goto** v", then

$$\nu_m(p, z, \bar{y}) \overset{\mathrm{df}}{=} \Big[(\chi[\mathrm{ext}(\bar{y}, z)] \to \mathrm{ext}(y_c, z+1) = v) \\ \wedge (\neg\chi[\mathrm{ext}(\bar{y}, z)] \to \mathrm{ext}(y_c, z+1) = i_{m+1}) \\ \wedge \bigwedge_{j<c} \mathrm{ext}(y_j, z+1) = \mathrm{ext}(y_j, z)\Big].$$

(iii) If $u_m =$ "**halt**", then

$$\nu_m(p, z, \bar{y}) \overset{\text{df}}{=} \bigwedge_{j \leq c} \text{ext}(y_j, z+1) = \text{ext}(y_j, z).$$

(iv) $$\nu(p) \overset{\text{df}}{=} \nu(p, \bar{y}) \overset{\text{df}}{=} \text{ext}(y_c, 0) = i_0 \wedge \forall z\Big[\Big(\bigvee_{m \leq n} \text{ext}(y_c, z) = i_m\Big) \wedge \bigwedge_{m \leq n} (\text{ext}(y_c, z) = i_m \to \nu_m(p, z, \bar{y}))\Big].$$

Now the definition of μ is the following: Let $w \in \omega$. Then

$$\mu(p, w) \overset{\text{df}}{=} \mu(p, w)(x_0, \ldots, x_{c-1}, x_w, \ldots, x_{w+c-1})$$
$$\overset{\text{df}}{=} \exists y_0 \cdots \exists y_c\Big[\bigwedge_{j<c} \text{ext}(y_j, 0) = x_j \wedge \nu(p, \bar{y}) \wedge \exists z\Big(\text{ext}(y_c, z) = i_n \wedge \bigwedge_{j<c} \text{ext}(y_j, z) = x_{w+j}\Big)\Big].$$

Note that $\mu(p, w) \in F_{\text{td}}$ since $(\forall m \leq n)\ \nu_m(p, z, \bar{y}) \in F_{\text{td}}$.

By this we have defined the function $\mu : P_d \times \omega \to F_{\text{td}}$.

Remark. For the 'meanings' of the formulas $\nu(p, \bar{y})$ and $\mu(p, w)$ see Fact 2 in Lemma 1.

(2) Now we define $\theta : \text{DF}_d \to F_{\text{td}}$ by recursion.

We shall use the function $\mu : P_d \times \omega \to F_{\text{td}}$ defined in step (1).

(i) Let $\varphi \in F_{\text{td}}$. Then

$$\theta(\varphi) \overset{\text{df}}{=} \varphi.$$

(ii) Let $\varphi, \psi \in \text{DF}_d$ and suppose that $\theta(\varphi)$ and $\theta(\psi)$ are already defined. Let $p \in P_d$.

The definition of $\theta(\Box(p, \psi))$:

Let w be the smallest element of ω such that $w \geq c$ and x_j does not occur in ψ for every $j \geq w$. Let $\bar{x} \overset{\text{df}}{=} \langle x_0, \ldots, x_{c-1}, x_w, \ldots, x_{w+c-1}\rangle$. Now

$$\theta(\Box(p, \psi)) \overset{\text{df}}{=} \theta(\Box(p, \psi))(x_0, \ldots, x_{c-1})$$
$$\overset{\text{df}}{=} \forall x_w \cdots \forall x_{w+c-1}\Big(\mu(p, w)(\bar{x}) \to \exists x_0 \cdots \exists x_{c-1}\Big(\bigwedge_{j<c} x_j = x_{w+j} \wedge \theta(\psi)\Big)\Big),$$

$$\theta(\varphi \wedge \psi) = \theta(\varphi) \wedge \theta(\psi),$$

$$\theta(\neg\varphi) = \neg\theta(\varphi),$$

$$\theta(\exists v\, \varphi) = \exists v\, \theta(\varphi) \quad \text{for every } v \in X \cup Y \cup Z.$$

By these we have defined a function θ on DF_d by induction. It is easy to check that $(\forall\varphi\in\mathrm{DF}_d)\ \theta(\varphi)\in F_{\mathrm{td}}$. □

Now we prove that θ satisfies condition $(*)$.

Lemma 1. *Let* $\varphi\in\mathrm{DF}_d$ *and* $\mathfrak{M}\in M_{\mathrm{td}}$. *Then* (i) *and* (ii) *below hold*:

(i) $\mathfrak{M}\models\varphi$ *iff* $\mathfrak{M}\models\theta(\varphi)$.

(ii) *Let* $g\in{}^{\omega}T$, $k\in{}^{\omega}D$, $r\in{}^{\omega}I$. *Then*

$$\mathfrak{M}\models\varphi[g,k,r] \text{ iff } \mathfrak{M}\models\theta(\varphi)[g,k,r].$$

Proof. It is enough to prove (ii).

Let $\mathfrak{M}\in M_{\mathrm{td}}$ and let $\langle g,k,r\rangle$ be a valuation of the variables into $\mathfrak{M}$, i.e. $g\in{}^{\omega}T$, $k\in{}^{\omega}D$, $r\in{}^{\omega}I$.

The following Fact 2 is easy to check (by inspecting Definition 11):

Fact 2. Let $p\in P_d$.

(1) $\mathfrak{M}\models\nu(p)[g,k,r]$ iff $\mathfrak{M}\models p[r]$ iff $\langle r_0,\ldots,r_c\rangle$ is a trace of p in $\mathfrak{M}$.

(2) Let $w\geqslant c$. Then $\mathfrak{M}\models\mu(p,w)[g,k,r]$ iff $\langle k_w,\ldots,k_{w+c-1}\rangle$ is a possible output of p with input $\langle k_0,\ldots,k_{c-1}\rangle$.

(3) Let $\psi\in\mathrm{DF}_d$. Then

$$\mathfrak{M}\models\theta(\Box(p,\psi))[g,k,r] \text{ iff } \mathfrak{M}\models\Box(p,\theta(\psi))[g,k,r].$$

Now we prove (ii) by induction on $\varphi\in\mathrm{DF}_d$. If $\varphi\in F_{\mathrm{td}}$, then (ii) holds by $\varphi=\theta(\varphi)$.

Let $\varphi,\psi\in\mathrm{DF}_d$, $v\in X\cup Y\cup Z$ and assume that (ii) holds for φ and ψ. Then (ii) holds for $\Box(p,\psi)$ by Fact 2(3). Then (ii) holds for $(\varphi\wedge\psi)$, $\neg\varphi$, $\exists v\,\varphi$ by $\theta(\varphi\wedge\psi)=\theta(\varphi)\wedge\theta(\psi)$, $\theta(\neg\varphi)=\neg\theta(\varphi)$ and $\theta(\exists v\,\varphi)=\exists v\,\theta(\varphi)$.

Hence by the definition of DF_d (Definition 9), (ii) holds for every $\varphi\in\mathrm{DF}_d$. □

Notation. Let $\mathrm{Th}\subseteq\mathrm{DF}_d$. Then $\theta[\mathrm{Th}]\overset{\mathrm{df}}{=}\{\theta(\varphi):\varphi\in\mathrm{Th}\}$.

Lemma 2. *The language* DL_d *is compact. I.e. let* $\mathrm{Th}\subseteq\mathrm{DF}_d$, $\varphi\in\mathrm{DF}_d$ *be such that* $\mathrm{Th}\models\varphi$. *Then* $H\models\varphi$ *for some finite* $H\subseteq\mathrm{Th}$.

Proof. Assume $\mathrm{Th}\models\varphi$. Then $\theta[\mathrm{Th}]\models\theta(\varphi)$ by Lemma 1. Since the classical 3-sorted language L_{td} is compact and since $\theta[\mathrm{Th}]\subseteq F_{\mathrm{td}}$, there is a finite $G\subseteq\theta[\mathrm{Th}]$ such that $G\models\theta(\varphi)$. Then there is a finite $H\subseteq\mathrm{Th}$ such that $G=\theta[H]$ and therefore $\theta[H]\models\theta(\varphi)$. By Lemma 1 then $H\models\varphi$. □

Definition 12 (classical proof concept $(\vdash,\mathrm{Prc})$ on F_{td}, [22]). $(\vdash,\mathrm{Prc})$ denotes the usual proof concept of the classical 3-sorted logic $L_{\mathrm{td}}=\langle F_{\mathrm{td}},M_{\mathrm{td}},\models\rangle$, see Definitions

10.14–10.27 of [22]. I.e.

$$\mathrm{Prc} \overset{\mathrm{df}}{=} \{\langle H, w, \varphi\rangle : H \subseteq F_{\mathrm{td}}, |H| < \omega, \varphi \in F_{\mathrm{td}} \text{ and } (\exists m \in \omega)\,[w \in {}^{m}F_{\mathrm{td}} \text{ and } w \text{ is a classical first order proof of } \varphi \text{ from } H \text{ in the sense of [22, Definition 10.24]}\}.$$

For any $\mathrm{Th} \subseteq F_{\mathrm{td}}$ and $\varphi \in F_{\mathrm{td}}$ we define

$$\mathrm{Th} \vdash \varphi \text{ iff } (\exists \langle H, w, \varphi\rangle \in \mathrm{Prc})\, H \subseteq \mathrm{Th}.$$

Note that $(\vdash, \mathrm{Prc})$ is a proof concept on F_{td} in the sense of Definition 10. □

Fact 3. $\vdash$ is a decidable proof concept (i.e. Prc is decidable), by Theorem 10.27 of [22].

Next we define our proof concept $\vdash^{\mathrm{N}}$.

Definition 13 (the proof concept $(\vdash^{\mathrm{N}}, \mathrm{Prn})$ on DF_d). By a $\vdash^{\mathrm{N}}$-proof of $\varphi \in \mathrm{DF}_d$ from $\mathrm{Th} \subseteq \mathrm{DF}_d$ we understand a sequence $\langle H, \langle \theta[H], w, \theta(\varphi)\rangle, \varphi\rangle$ such that $H \subseteq \mathrm{Th}$ and $\langle \theta[H], w, \theta(\varphi)\rangle$ is a classical proof of $\theta(\varphi)$ from $\theta[H]$. In more detail:

$$\mathrm{Prn} \overset{\mathrm{df}}{=} \{\langle H, \langle \theta[H], w, \theta(\varphi)\rangle, \varphi\rangle : H \subseteq \mathrm{DF}_d, |H| < \omega, \varphi \in \mathrm{DF}_d \text{ and } \langle \theta[H], w, \theta(\varphi)\rangle \in \mathrm{Prc}\}.$$

Let $\mathrm{Th} \subseteq \mathrm{DF}_d$, $\varphi \in \mathrm{DF}_d$. Then we define

$$\mathrm{Th} \vdash^{\mathrm{N}} \varphi \text{ iff } (\exists H \subseteq \mathrm{Th})(\exists w)\langle H, w, \varphi\rangle \in \mathrm{Prn}.$$

Note that $(\vdash^{\mathrm{N}}, \mathrm{Prn})$ is a proof-concept on DF_d in the sense of Definition 10. □

Lemma 3. $\vdash^{\mathrm{N}}$ *is a decidable proof concept.*

Proof. We have to prove that $\mathrm{Prn} \subseteq (\mathrm{DF}_d)^*$ is decidable. Let $\pi \in (\mathrm{DF}_d)^*$ be arbitrary. The algorithm of deciding whether $\pi \in \mathrm{Prn}$ or not goes as follows:

If π is not a triple, then $\pi \notin \mathrm{Prn}$. Assume $\pi = \langle H, w, \varphi\rangle$. By definition of $(\mathrm{DF}_d)^*$ (and by our convention that we identify X^* with $\{H : H \in X, |H| < \omega\}^*$) we have that H is finite. We also have that DF_d and Prc are decidable.

If $H \nsubseteq \mathrm{DF}_d$ or $\varphi \notin \mathrm{DF}_d$ or $w \notin \mathrm{Prc}$, then $\pi \notin \mathrm{Prn}$. Assume $H \subseteq \mathrm{DF}_d$, $\varphi \in \mathrm{DF}_d$ and $w \in \mathrm{Prc}$. Since $w \in \mathrm{Prc}$ we have that $w = \langle K, v, \psi\rangle$ for some finite $K \subseteq F_{\mathrm{td}}$ and for some $\psi \in F_{\mathrm{td}}$.

Compute $\theta[H]$. Since θ is computable, computing $\theta[H]$ terminates in finite time. Compute $\theta(\varphi)$ similarly.

If $\theta[H] \neq K$ or $\theta(\varphi) \neq \psi$, then $\pi \notin \mathrm{Prn}$, else $\pi \in \mathrm{Prn}$. □

Lemma 4 (completeness of the logic $\langle \mathrm{DL}_d, (\vdash^{\mathrm{N}}, \mathrm{Prn})\rangle$). *Let* $\mathrm{Th} \subseteq \mathrm{DF}_d$ *and* $\varphi \in \mathrm{DF}_d$ *be arbitrary. Then*

$$\mathrm{Th} \models \varphi \textit{ iff } \mathrm{Th} \vdash^{\mathrm{N}} \varphi.$$

Proof. (1) Assume $\mathrm{Th} \models \varphi$. Then $\theta[\mathrm{Th}] \vdash \theta(\varphi)$ by Lemma 1. Then by Gödel's completeness theorem (see [22, Theorem 11.20]) $\theta[\mathrm{Th}] \vdash \theta(\varphi)$, i.e. there is a classical first order proof $\langle K, w, \theta(\varphi)\rangle \in \mathrm{Prc}$ of $\theta(\varphi)$ from $\theta[\mathrm{Th}]$. Then $K \subseteq \theta[\mathrm{Th}]$ is finite, i.e. $K = \theta[H]$ for some finite $H \subseteq \mathrm{Th}$. Then $\langle H, \langle \theta[H], w, \theta(\varphi)\rangle, \varphi\rangle \in \mathrm{Prn}$ is an $\vdash^{\mathrm{N}}$-proof of φ from Th. Thus $\mathrm{Th} \vdash^{\mathrm{N}} \varphi$.

(2) Assume $\mathrm{Th} \vdash^{\mathrm{N}} \varphi$. Then there is $\langle H, \langle \theta[H], w, \theta(\varphi)\rangle, \varphi\rangle \in \mathrm{Prn}$ for some finite $H \subseteq \mathrm{Th}$ such that $\langle \theta[H], w, \theta(\varphi)\rangle \in \mathrm{Prc}$. By soundness of classical first order logic (Monk [22, Theorem 11.8]) $\theta[H] \models \theta(\varphi)$. Then by Lemma 1 $H \models \varphi$, and therefore $\mathrm{Th} \models \varphi$. □

Lemmas 3 and 4 prove Theorem 2. Moreover by Fact 1, Lemma 3, and Lemma 4 we proved Theorem 1, too. □

By a *logic* we understand a pair $\langle L, (\vdash, \mathrm{Pr})\rangle$ where $L = \langle F, M, \models\rangle$ is a language in the sense of Section 2 and $(\vdash, \mathrm{Pr})$ is a proof concept for L in the sense of Definition 10.

The logic $\langle L, (\vdash, \mathrm{Pr})\rangle$ is said to be *complete* iff [$(\vdash, \mathrm{Pr})$ is a decidable proof concept and for all $\mathrm{Th} \subseteq F$ and $\varphi \in F$ we have ($\mathrm{Th} \models \varphi$ iff $\mathrm{Th} \vdash \varphi$)].

We define First order *Dynamic Logic* of type d to be the logic $\langle \mathrm{DL}_d, (\vdash^{\mathrm{N}}, \mathrm{Prn})\rangle$ where the proof concept $(\vdash^{\mathrm{N}}, \mathrm{Prn})$ was defined in Definition 13. In Part II we shall often say sloppily "the logic DL_d". In these cases we shall always mean to say "the logic $\langle \mathrm{DL}_d, (\vdash^{\mathrm{N}}, \mathrm{Prn})\rangle$".

About the axiomatic or Hilbert style version of the proof concept $(\vdash^{\mathrm{N}}, \mathrm{Prn})$ see the beginning of Section 5 in Part II.

By Theorem 2 our First order Dynamic Logic $\langle \mathrm{DL}_d, (\vdash^{\mathrm{N}}, \mathrm{Prn})\rangle$ is complete.

References

[1] H. Andréka, L. Csirmaz, I. Németi and I. Sain, More complete logics for reasoning about programs Preprint, Math. Inst. Hung. Acad. Sci. (1980).

[2] H. Andréka and I. Németi, A characterization of Floyd provable programs, in: *Mathematical Foundations of Computer Science MFCS '81* (Springer, Berlin, 1981). Also: Preprint, Math. Inst. Hung. Acad. Sci. No. 8 (1978).

[3] H. Andréka and I. Németi, Completeness of Floyd method w.r.t. nonstandard time models, Seminar Notes, Math. Inst. Hung. Acad. Sci.-SZKI (1977, in Hungarian). Abstracted in: *Bull. Section of Logic, Wroclaw*, **7** (1978) 115–120.

[4] H. Andréka, I. Németi and I. Sain, Completeness problems in verification of programs and program schemes, in: J. Becvár, Ed., *Mathematical Foundations of Computer Science 1979*, Lecture Notes in Computer Science **74** (Springer, Berlin, 1979) 208–218.

[5] H. Andréka, I. Németi and I. Sain, Henkin-type semantics for program schemes to turn negative results to positive, in: L. Budach, Ed., *Fundamentals of Computation Theory FCT'79* (Akademie, Berlin, 1979) 18–24.

[6] H. Andréka, I. Németi and I. Sain, Program verification within and without logic, *Bull. Section of Logic, Wroclaw* **8** (1979) 124–130.
[7] F. Berman, A completeness technique for *D*-axiomatizable semantics, *Proc. 11th Annual ACM Symposium on Theory of Computing*, Atlanta, GA (1979) 160–166.
[8] F. Berman, Syntactic and semantic structure in Propositional Dynamic Logic, Technical Report No. 79-07-05, Department of Computer Science, University of Washington, Seattle 98195 (1979).
[9] B. Biró, On the completeness of program verification methods, *Bull. Section of Logic, Wroclaw* **10**(2) (1981).
[10] K.A. Bowen, *Model Theory for Modal Logic* (Kripke *Models for Modal Predicate Calculi*), Synthese Library **127** (Reidel, Dordrecht, 1979).
[11] R. Burstall and J. Darlington, A system which automatically improves programs, *Proc. 3rd IJCAI* (S.R.I., 1973) 537–542.
[12] C.C. Chang and H.J. Keisler, *Model Theory* (North-Holland, Amsterdam, 1973).
[13] B. Courcelle and I. Guessarian, On some classes of interpretations, *J. Comput. System. Sci.* **17** (1978) 388–413.
[14] L. Csirmaz, Structure of program runs of nonstandard time, *Acta Cybernet.* **4** (1980) 325–331.
[15] L. Csirmaz, Programs and program verifications in a general setting, *Theoret. Comput. Sci.* **16** (1981) 199–210.
[16] D. Gallin, *Intensional and Higher Order Modal Logic* (North-Holland/American Elsevier, New York, 1978).
[17] T. Gergely and M. Szöts, Model theoretic investigations in programming theory, *Acta Cybernet.* **4** (1979) 45–57.
[18] T. Gergely and L. Úry, Specification of program behaviour through explicit time considerations, in: S.H. Lavington, Ed., *Information Processing 80* (North-Holland, Amsterdam, 1980) 107–111.
[19] D. Harel, *First-Order Dynamic Logic*, Lecture Notes in Computer Science **68** (Springer, Berlin, 1979).
[20] Y. I. Ianov, The logical schemes of algorithms, in: *Problems of Cybernetics Vol. 1* (Pergamon Press, New York, 1960) 82–140.
[21] Z. Manna, *Mathematical Theory of Computation* (McGraw-Hill, New York, 1974).
[22] J.D. Monk, *Mathematical Logic* (Springer, Berlin, 1976).
[23] I. Németi, Connections between algebraic logic and initial algebra semantics of CF languages, in: B. Dömölky and T. Gergely, Eds., *Mathematical Logic in Computer Science*, Colloquia Mathematica Societatis János Bolyai **26** (North-Holland, Amsterdam) 25–83, 561–605.
[24] R. Parikh, The completeness of propositional dynamic logic, in: J. Winkowski, Ed., *Mathematical Foundations of Computer Science 1978*, Lecture Notes in Computer Science **64** (Springer, Berlin, 1978) 403–415.
[25] V.R. Pratt, A practical decision method for propositional dynamic logic, *Proc. 10th Annual ACM Symposium on Theory of Computing*, San Diego, CA (1978) 326–337.
[26] V.R. Pratt, Models of program logics, *Proc. 20th IEEE Conference on Foundations of Computer Science*, San Juan, PR (1979).
[27] V.R. Pratt, Flowgraph logic and the elimination of Kleene elimination, MIT Preprint No. 4/17/80 (1980).
[28] I. Sain, First order dynamic logic with decidable proofs and workable model theory, in: *Fundamentals of Computation Theory FCT '81* (Springer, Berlin, 1981).
[29] I. Sain, There are general rules for specifying semantics: Observations on Abstract Model Theory, *CL & CL—Comput. Linguist. Comput. Lang.* **13** (1979) 251–282.
[30] K. Segerberg, A completeness theorem in the modal logic of programs, abstract, *Notices Amer. Math. Soc.* **24** (6) (1977) A-552.
[31] K. Segerberg, Applying modal logic, *Studia Logica* **39** (1980) 275–296.
[32] J. Stavi, Compactness properties of infinitary and abstract languages, in: A. Macintyre, L. Pacholski and J. Paris, Eds., *Logic Colloquium '77* (North-Holland, Amsterdam, 1978) 263–275.