ON THE STRENGTH OF TEMPORAL PROOFS[1]

Hajnal Andréka, István Németi, and Ildikó Sain
Mathematical Institute of the Hungarian Academy of Sciences
Budapest, Pf.127, H–1364, Hungary

In this paper we investigate the reasoning powers or proof theoretic powers of various established temporal logics used in Computer Science. In the first part we concentrate on provability of various program properties while in the second one we investigate provability of temporal formulas in general. In the first part we consider both deterministic and nondeterministic programs.

Our investigations are twofold:
(1) *compare* the reasoning powers of the various logics, and
(2) *characterize* their reasoning powers.

The investigations in (2) are often called *completeness* issues, because a good characterization amounts to finding a nice and mathematically transparent semantics w.r.t. which our logic is complete, cf. ABADI [2] and [10]. In doing (2), we follow the methodology called Correspondence Theory in philosophical logic (see Chap.II.4 of [10]) which was first elaborated for temporal logics of programs in the 1978 version of SAIN [23] (cf. also [5], both papers based on the Computer Science temporal logics in [4]), in the framework called time oriented Nonstandard Logics of Programs (NLP). Same is used in ABADI [1], [2]. In particular, the semantics denoted as "$\vdash_0 P(\dots)$" by ABADI was first introduced as "$(Ind+Tord) \models$" in the above quoted NLP literature, and will play a central rôle herein, too. Among others, we will obtain new strong (hereditarily in a sense) incompleteness results w.r.t. this semantics for proof systems of ABADI–MANNA [3] and MANNA–PNUELI [18] respectively. No number of new axioms, but a single new modality can eliminate this incompleteness.

§1. Time oriented NLP, and the first characterization result

Time oriented NLP is a three–sorted classical first–order logic, the sorts of which are the time scale T, the data domain D, and a sort I consisting of some functions from T into D. We think of the elements of I as time sequences, i.e. sequences indexed by the elements of T. In more detail, a model of time oriented NLP is a triple $\mathfrak{M} = \langle \mathbf{T}, \mathbf{D}, I \rangle$ where $\mathbf{T} = \langle T, 0, suc, \leq, +, \cdot \rangle$ is called the time structure of $\mathfrak{M}$, $\mathbf{D}$ is the data structure (or data domain) of $\mathfrak{M}$, and $I \subseteq {}^T D$ (${}^T D$ denotes the set of all functions from T into D.) The intuition comes from the *standard model* $\langle \langle \omega, 0, suc, \leq, +, \cdot \rangle, \mathbf{D}, {}^\omega D \rangle$ of NLP, where ω is the set of all natural numbers, and the elements of I are indeed ω–sequences. For our purposes, however, the arbitrary models like $\mathfrak{M}$ above are more important than the standard one.

Let p be a (possibly nondeterministic) blockdiagram program using only one variable x. For $y \in I$ we say that y is an execution sequence of p if the sequence $\langle y(0), y(suc(0)), \dots, y(t), \dots \rangle_{t \in T}$ is an execution of p in the usual sense. Now a partial correctness assertion, pca from now on, $\{\varphi(x)\}p\{\psi(x)\}$ is valid in $\mathfrak{M}$ iff for every execution sequence $y \in I$ of p, whenever $\varphi(y(0))$ holds in $\mathbf{D}$ and $y(t)$ is

[1]This work has been supported by the Hungarian National Foundation for Scientific Research Grant No 1810.

a terminating state (or possible output) of p, then we also have that $\psi(y(t))$ holds. The case when p uses more variables is similar. (Total correctness and other kinds of statements about programs are formalized in time oriented NLP similarly, cf. §5 way below.)

The *formulas* of time oriented NLP are (basically) the usual three-sorted first-order formulas of the language of $\mathfrak{M}$. Let Ax be a set of such formulas of time oriented NLP. Then

$$Ax \models \{\varphi\}p\{\psi\}$$

is defined to hold iff $\{\varphi\}p\{\psi\}$ is valid in every model $\mathfrak{M}$ of Ax. We say that $\{\varphi\}p\{\psi\}$ follows from Ax (in time oriented NLP) iff $Ax \models \{\varphi\}p\{\psi\}$ holds.

A characterization of a fixed program verification method, say $\vdash^{FH}$, consists of finding a set Ax of (first-order) formulas of the many-sorted language of time oriented NLP, and proving that *the correctness of a program is provable by* $\vdash^{FH}$ *iff it follows from* Ax *in time oriented NLP.* An example for such a characterization of Floyd-Hoare method $\vdash^{FH}$ is the theorem saying that a pca is Floyd-Hoare provable iff it follows from the axiom system Ind_{qf} of time oriented NLP (see [27], [9]). Ind_{qf} is a restricted T-induction. More precisely, let us recall that the time scale $\mathbf{T}$ has a distinguished element $0 \in T$ and a nexttime function or successor $suc : T \longrightarrow T$. Now, for *any* formula $\varphi(z)$ of time oriented NLP, the full induction schema Ind postulates

$$(\varphi(0) \wedge (\forall z \in T)[\varphi(z) \rightarrow \varphi(suc(z))]) \rightarrow (\forall z \in T)\varphi(z).$$

Note that in this Ind, φ may speak about the whole of $\mathfrak{M}$ and not only $\mathbf{T}$, and may contain parameters from *any* sorts of $\mathfrak{M}$. So our Ind is much stronger than the usual one in which φ is allowed to talk about the structure $\mathbf{T}$ only. Now, $Ind_{qf} \subset Ind$ postulates induction for exactly those formulas $\varphi(z)$ of time oriented NLP which contain no quantifiers of sort T. The above mentioned characterization of the Floyd-Hoare method $(\vdash^{FH})$ in the notational system of the present paper reads as

$$\vdash^{FH} \equiv_{\Box} Ind_{qf} .$$

This formula abbreviates the claim that for any pca $\{\varphi\}p\{\psi\}$ we have:

$$\vdash^{FH} \{\varphi\}p\{\psi\} \quad \Longleftrightarrow \quad Ind_{qf} \models \{\varphi\}p\{\psi\} .$$

We will return to the nondeterministic and concurrent cases in §6 below. All the results herein carry over to the nondeterministic- and concurrent cases under the assumption of the existence of a clock in the sense of ABADI [1], [2] or PARIKH [21] or §6 here.

§2. Temporal logics of programs, and further characterizations

We will use a first-order multimodal (actually temporal) logic with five modalities *First*, *Next*, *Always*, *"Always-in-the-future"*, and *"Always-in-the-past"*. We will abbreviate the last three as *Alw*, *Afu*, and *Apa*. The intuitive meaning of

First φ is that φ is true at the first time instant.[2] Using our three–sorted models $\mathfrak{M}$ introduced for time oriented NLP in §1 above, *First* φ says that φ is true at time 0. Similarly, for $t \in T$, *Next* φ is true at t iff φ is true at $suc(t)$. *Alw* φ is true at t iff for all $t_1 \in T$, φ is true at t_1. *Afu* φ is true at t iff for all $t_1 \in T$ with $t_1 \geq t$, φ is true at t_1. *Apa* φ is true at t iff for all $t_1 \leq t$, φ is true at t_1. Finally, $P(y)$ with $y \in I$ is true at t iff $P(y(t))$ holds in $\mathbf{D}$. $\mathfrak{M} \models \varphi$ for a temporal formula φ iff φ is true at every $t \in T$.

A nonlogical symbol is called flexible (opposite of "rigid") if it is allowed to change in time. Unless otherwise specified, the only flexible symbols we allow are constants $y_0, y_1, \ldots y_n, \ldots$. However, in some of our results we will allow flexible predicates etc. too. Throughout, y_i and x_i denote flexible constants and rigid variables respectively.

Axiomatization of temporal logic:

Consider axioms $(A1, 2)$ and rules $(A3-5)$ below.

$(A1)$ For any propositional temporal schema ψ valid in the standard models $\langle \omega, \ldots \rangle$, all (first–order) instances of ψ belong to $(A1)$.

Remark: It is known that $(A1)$ is decidable, and many finite axiomatizations are available for $(A1)$, Cf. [12], GOLDBLATT [13].

$(A2)$ For every temporal formula φ, if φ is valid in every one of the models $\mathfrak{M}$ of time oriented NLP then φ is in $(A2)$.

Remark: $(A2)$ can be replaced with the HILBERT–style axioms: $\{(\varphi \leftrightarrow \boxdot \varphi)$ for every modality $\boxdot$ (i.e. $\boxdot \in \{First, Next, Alw, Afu, Apa\}$) if φ contains no flexible symbols; $(\forall x \boxdot \varphi \leftrightarrow \boxdot \forall x \varphi)$ for every modality $\boxdot$; $\varphi \rightarrow \varphi(x/\tau)$ for any term τ such that the substitution $x \mapsto \tau$ does not create new bound occurrences of variables or new occurrences of flexible symbols in the scope of modalities in φ; all (temporal instances of all) axiom schemata of the axiomatization on p.157 of [16] of classical first–order logic (other axiomatizations work too, but one has to avoid possible substitution rules)$\}$,

$(A3)$ $\quad \{\varphi, \varphi \rightarrow \psi\} \vdash \psi$,

$(A4)$ $\quad \varphi \vdash \forall x\, Alw\, \varphi$,

$(A5)$ $\quad \{First\, \varphi,\ \varphi \rightarrow Next\, \varphi\} \vdash \varphi$.

Derivability with $(A1-5)$ is denoted by $\vdash_{SFP}$, derivability with $(A1-5)$ in the fragment not containing *Apa* is $\vdash_{SF}$, same in fragment not containing either *Apa* or *Afu* is $\vdash_S$, and same in fragment containing *First* and *Next* only is $\vdash_0$. (Here the indices S, F, P refer to *Sometime* which is interdefinable with *Alw*, *Afu*, *Apa* being allowed in addition to *First* and *Next* to occur in the formulas.) Note that $\vdash_S$, $\vdash_{SF}$, $\vdash_{SFP}$ are frequently used established temporal logics.

For a pca $\{\varphi\}p\{\psi\}$ we use its usual temporal representation $temp(\{\varphi\}p\{\psi\})$ which is in the fragment of $\vdash_0$ (i.e. temporal formulas using *First* and *Next* only, cf. e.g.

[2] Our modality *First* might look unorthodox. However, it is expressible in the temporal logics in PNUELI [22], MANNA–PNUELI [19] or LICHTENSTEIN–PNUELI–ZUCK [17], namely *First*φ is equivalent with $\langle Future\rangle\langle Past\rangle(\varphi \wedge \neg \ominus TRUE)$ or equivalently $\langle Future\rangle\langle Past\rangle(\varphi \wedge \ominus FALSE)$. Here "$\langle Future\rangle\langle Past\rangle$" expresses *Sometime* and $\ominus$ is *strong*– while $\ominus$ is *weak "previously"*. A similar remark applies to the temporal logic *LinDisc* on p.64 of GOLDBLATT [13]. Similarly, in any temporal logic to which Exercise 6.7 (p.44) of [13] applies, the above used $\langle Future\rangle$ and $\ominus$ are expressible, hence *First* is so.

PNUELI [22]). We do not distinguish the original pca ρ from $temp(\rho)$, hence we write $\vdash_S \{\varphi\}p\{\psi\}$ for $\vdash_S temp(\{\varphi\}p\{\psi\})$.

PROPOSITION 1. $\vdash^{FH} \equiv_{\square} \vdash_0$. *This extends to nondeterministic programs, too.* ∎

THEOREM 2. **None** *of the extensions* $\vdash_0 \mapsto \vdash_S \mapsto \vdash_{SF} \mapsto \vdash_{SFP}$ *is conservative; i.e. there are formulas* φ_0, φ_1, φ_2 *in the fragment of* $\vdash_0$ *such that* $\nvdash_0 \varphi_0$ *but* $\vdash_S \varphi_0$, $\nvdash_S \varphi_1$ *but* $\vdash_{SF} \varphi_1$ *etc.* ∎

Our first group of characterizations of program verifying powers is:

THEOREM 3.
$$\vdash^{FH} \equiv_{\square} Ind_{qf}$$
$$\vdash^{IAM} \equiv_{\square} Ind$$
$$\vdash^{PNU} \equiv_{\square} (Ind + \textit{Time is linearly ordered}) \equiv_{\square} \vdash_{SFP} .$$

*Here the time oriented NLP axiom "*Time is linearly ordered*" is equivalent with postulating the full first-order theory* $Th(\langle\omega, 0, suc, \leq\rangle)$ *of the standard structure* $\langle\omega, 0, suc, \leq\rangle$ *for the time sort (or time scale)* **T** *of our models* $\mathfrak{M}$. ∎

COROLLARY 4. $\vdash_{SF} \equiv_{\square}$ ($\vdash_{SFP}$ *expanded with "Until" and "Since"*). *I.e.* $\vdash_{SF} \equiv_{\square}$ *"the strongest possible temporal logic based on linear discrete ordering of time".* ∎

Tord abbreviates "Time is linearly ordered", from now on.

§3. Comparing program verifying powers

From the point of view of proving program properties, $\vdash_S$ and $\vdash_{SF}$ are the same as the established program verification methods known as *Intermittent Assertions Method* (or *Sometime Method*) and PNUELI's temporal method respectively.

Already from the point of view of proving *deterministic pca's only*, Floyd–Hoare method is strictly weaker than $\vdash_S$ which is strictly weaker than $\vdash_{SF} \equiv_{\square} \vdash_{SFP}$ which in turn is strictly weaker than some new methods to be introduced and discussed in §4 below. In symbols

THEOREM 5. $\vdash^{FH} <_{\square} \vdash_S <_{\square} \vdash_{SF} <_{\square}$ (*certain new methods*)
where e.g. $\vdash^{FH} <_{\square} \vdash_S$ *means that strictly more deterministic pca's are provable by* $\vdash_S$ *than by* $\vdash^{FH}$. *(I.e. if* $\equiv_{\square}$ *would have been defined as* $[\leq_{\square}$ *and* $\geq_{\square}]$ *then* $<_{\square}$ *would be* $[\leq_{\square}$ *and not* $\geq_{\square}]$*).* ∎

At this point we note that the symbols $\equiv_{\square}$ and $<_{\square}$ are applicable between any formalisms $\vdash_1$ and $\vdash_2$ which are suitable for proving pca's. So HAREL's axiomatization of dynamic logic or any other logic of programs can take the place of $\vdash_i$ in $\vdash_1 \equiv_{\square} \vdash_2$ or $\vdash_1 <_{\square} \vdash_2$.

In connection with the differences in proof theoretic (or program verifying) power discussed so far, the following question of practical relevance comes up:

"What happens if the data domain **D** is rich enough to encode finite sequences with single elements?" More precisely, what we assume is that the data theory (or specification) forces the data domain to be such. Examples for such "rich" data theories are Peano's Arithmetic, the specification of LISP, finite (or arbitrary) set theory with or without urelements. The answer to this question is that if the data theory ensures codability of finite sequences then Floyd–Hoare method $\vdash^{FH}$ becomes as strong as Pnueli's $\vdash_{SFP}$ which in turn remains still strictly weaker than the new methods mentioned above. In symbols

THEOREM 6.

$$\left(\vdash^{FH} \equiv_{\square} \vdash_{S} \equiv_{\square} \vdash_{SFP} <_{\square} \ (certain\ new\ methods)\right) \Big/ \begin{pmatrix} modulo\ Peano's \\ arithmetic\ for\ data \end{pmatrix} . \ \blacksquare$$

Thm.5.1.(iv) of SAIN [26] p.312 contains more information on the "$<_{\square}$ *(certain new methods)*" part above and Thm.5.1(vi) [26] is the *total correctness* version of the above theorem.

At this point a further characterization result can be presented. Namely let $\vdash^{HAREL}$ be the inference system of (standard) Dynamic Logic as presented in HAREL [15] and also in Def.10 on p.493 of SAIN [24]. Then,

THEOREM 7.

$$\left(\vdash^{HAREL} \equiv_{\square} \ (Ind + Peano's\ arithmetic\ for\ Time)\right) \Big/ \begin{pmatrix} modulo\ Peano's \\ arithmetic\ for\ data \end{pmatrix} . \ \blacksquare$$

This follows from Thm.5 on p.496 of [15] together with Thm.5.1(v) of SAIN [26] p.312. Actually Thm.5 of [26] gives a more general characterization of $\vdash^{HAREL}$ too, namely w.r.t. all statements of programs expressible in standard Dynamic Logic. Instead of recalling that characterization in full detail, we mention that it uses besides "*(Ind + Time is linearly ordered)*" a restricted form of Ex defined in §4 below together with three–sorted induction on data (i.e. the same as Ind but for $\mathbf{D}$ instead of $\mathbf{T}$) which is sometimes called structural induction. We note that full Ex would be too strong. We will return to this in §4.

§4. Using NLP, it is easy to construct **program verification methods strictly stronger than Pnueli's temporal logics of programs**. Some of these strong methods have been defined in terms of (usual) *temporal logics*, too (see ANDRÉKA–NÉMETI–SAIN [6]). An example for a program the partial correctness of which is provable by such a strong method but *not provable by Pnueli's method* is: *a program verifier for LISP programs*. A more mundane example is a proof checker for theorems about LISP or about Peano's Arithmetic. Some of these new methods remain strictly stronger than Pnueli's one *even if the data theory ensures codability of finite sequences* (e.g. if it contains Peano's Arithmetic).

In more detail, $(Ind + Tpa + Ex)$ is a set of axioms in the three–sorted first–order language of time oriented NLP.: Ex, "existence axioms", postulates the existence of those elements of I which are definable by first–order three–sorted formulas. In traditional logic Ex is usually called comprehension schema, see §D.4.5 (p.937) of [7]. More concretely, if $\mathfrak{M} = \langle \mathbf{T}, \mathbf{D}, I \rangle$ and $\varphi(z,x)$ is a formula (in the first–order language of $\mathfrak{M}$) with $\mathfrak{M} \models (\forall z \in T)(\exists x \in D)\varphi(z,x)$ then Ex postulates the existence of a $y \in I$ with $\mathfrak{M} \models (\forall z \in T)\varphi(z, y(z))$. Further, Tpa abbreviates "Peano's axioms for the time scale $\mathbf{T}$ expanded with $+$ and $\cdot$ ". (Note that "Peano's axioms for the data domain $\mathbf{D}$", in short "Peano's arithmetic for data", is disjoint from Tpa since it speaks about a different sort of $\mathfrak{M}$.

THEOREM 8.

(a) $\vdash_{SFP} <_{\square} (Ind + Tpa) <_{\square} (Ind + Tpa + Ex)$, *and*

(b) $\Big((Ind + Tpa) <_{\square} (Ind + Tpa + Ex)\Big) \Big/ \binom{\text{modulo Peano's}}{\text{arithmetic for data}}$.

One concludes that $(Ind + Tpa + Ex)$ *is strictly stronger than the strongest proof system* T_2 *studied in* ABADI [1],[2] *because theorems therein state that* $\vdash_{T_2} \leq_{\square} (Ind + Tpa)$. ∎

Outline of proof: Let us consider a theorem prover program p deriving consequences from Peano's axioms. Now, our pca says that p will never derive the formula $x \neq x$. If we wanted a Floyd–Hoare proof for this pca, we would be looking for an "inductive assertion" χ. For our present pca, a natural inductive assertion is the following: First we fix a model say $\mathbf{N}$ of Peano's arithmetic, and then χ says that the formula derived by p in the actual (or "present") step is valid in our fixed model $\mathbf{N}$. This χ will be true in the zeroth step (since $\mathbf{N} \models$ *Peano's axioms*) and if χ is true at time $t \in T$ then it will be easily seen to be true at $t + 1$. Therefore by induction, this χ would be suitable for proving our pca.

However, this χ is *not* expressible in the language of our data domain, moreover it is not even expressible even in $(Ind + Tpa + \text{"}Peano's\ axioms\ for\ data\text{"})$, because of TARSKI's theorem on the undefinability of truth. Intuitively, the reason for this is that expressing χ assumes defining a model, say $\mathbf{N}$, but $\mathbf{N}$ is an essentially infinite object while the elements of our data domain are essentially finite. Even the elements of the models of $(Ind + Tpa + \text{"}Peano's\ axioms\ for\ data\text{"})$ are "internally" finite, i.e. logically they behave like finite objects. Since $(Ind + Tpa + \text{"}Peano's\ axioms\ for\ data\text{"})$ ensures the existence of these finite objects only, we cannot define (any element that would be big enough to code) $\mathbf{N}$ in the framework of this theory. TARSKI's theorem adds that this limitation cannot be sidestepped by some "clever trick". This inability of expressing χ leads to unprovability of our pca in $(Ind + Tpa + \text{"}Peano's\ axioms\ for\ data\text{"})$, hence it is also unprovable in $\vdash_{SFP}$. We will return to this unprovability a little bit later.

Our main point here is that the above discussed χ *is* expressible in $(Ind + Tpa + Ex)$ because Ex ensures the availability of infinite objects, from which we can construct models. Namely, the functions $y \in I$, mapping T into D can be used as characteristic functions of subsets of T. These subsets can be infinite even "internally" (e.g. the set of odd elements of T is easily codable by a $y : T \longrightarrow D$ and it is an infinite set from all possible points of view). Therefore, Ex enables us to construct (or "code") a model of Peano's arithmetic from elements of I, to prove that this model indeed exists and satisfies Peano's axioms etc. In short, χ is expressible in $(Ind + Tpa + Ex)$ and therefore our pca is provable. The details are worked out both for provability from $(Ind + Tpa + Ex)$ and for unprovability from $(Ind + Tpa + \text{"}Peano's\ axioms\ for\ data\text{"})$ in §V.1 of SAIN [25].

As promised above, concerning unprovability from $(Ind + Tpa + \text{"}Peano's\ axioms\ for\ data\text{"})$: A proof of our pca from this theory would imply provability of the consistency of Peano's arithmetic from $(Ind + Tpa + \text{"}Peano's\ axioms\ for\ data\text{"})$. However, this theory is equiconsistent with Peano's arithmetic (an easy exercise), hence the assumption $(Ind + Tpa + \text{"}Peano's\ axioms\ for\ data\text{"}) \vdash$ (*our pca*) would imply provability of

the consistency of a theory from itself, contradicting GÖDEL's incompleteness theorem. See [25] for details.

The second part of (a) follows from (b). In connection with the first part of (a), we note that a pca distinguishing $\vdash_{SFP}$ from $(Ind{+}Tpa)$ expresses the partial correctness of a theorem prover which, while being nontrivial, is inherently simpler than one for LISP or Peano's arithmetic. See BIRÓ–SAIN [8] for the details. ∎

More material on the subject of the present section is found in BIRÓ–SAIN [8], §5 of [26], [6], [25], HÁJEK [14].

§5. Eventualities, total correctness:

One can treat provability of *eventualities* (like total correctness assertions) in NLP, as well as other kinds of statements about programs, see e.g. SAIN [26],[24]. The following result is an illustration for this:

THEOREM 9. *From the point of view of total correctness assertions, Intermittent Assertions Method is strictly weaker than Pnueli's method. But the difference between the powers of these two methods disappears if we assume that the data theory ensures codability of finite sequences (e.g. if it contains Peano's Arithmetic).* ∎

A *characterization* of Intermittent Assertions Method from the point of view of total correctness is given in Theorems 2.7 and 4.4 of [26] under the assumption that the data theory (or equivalently specification) contains Peano's axioms (postulated for the data sort, of course). Cf. also p.286 lines 16–17 of [26]. It seems likely that the condition that the data theory has to contain Peano's arithmetic can be eliminated from the quoted characterization in [26] if we change the frame of characterization slightly. Namely, the assertion "the program p terminates" was represented (in the formalism of time oriented NLP) in [26] by saying that p *has an execution sequence which terminates.* If instead, we use the statement saying "*every execution sequence of* p *terminates*", we get a slightly different representation of total correctness assertions. We conjecture that under this new representation the characterizations in [26] can be improved.

§6. Concurrency, nondeterminism, fairness, and clocks

The characterization $\vdash^{FH} \equiv_{\Box} Ind_{qf}$ in Theorem 1 carries over to concurrent and nondeterministic programs without any further assumptions (like clocks).

THEOREM 10.

(i) $\vdash_S <_{\Box}$ *Ind for nondeterministic pca's.*

(ii) *There is a nondeterministic pca* ρ *such that*
$\vdash_{SFP}$ "ρ *holds for* **fair** *executions" but*
$\nvdash_{SF}$ "ρ *holds for* **fair** *executions".*

That is, **fairness** *separates* $\vdash_{SF}$ *and* $\vdash_{SFP}$. ∎

THEOREM 11. $\vdash_{SFP} \equiv (Ind{+}Tord)$ *for* **all** *properties of deterministic programs. This is* **not** *true for* $\vdash_{SF}$ *in place of* $\vdash_{SFP}$. ∎

Recall from ABADI [1], [2], PARIKH [21] that a clock is a temporal formula $\gamma(\bar{x})$ satisfying

$$C(\gamma) \stackrel{\text{def}}{=} Alw(\exists\, \bar{x}\gamma(\bar{x}) \wedge [\gamma(\bar{x}) \rightarrow Next\ Afu\ \neg\gamma(\bar{x})]).$$

So a clock γ never "shows the same time" in two different time instances t and t_1. A weak clock, formalized as $Cw(\gamma)$, is permitted to show the same time in t and t_1 but then t and t_1 should not be distinguishable by atomic formulas or by $Next\ \gamma(\overline{x})$. Assume $\gamma(\overline{x})$ is of form $\overline{y} = \overline{x}$. Then $Cw(\gamma)$ postulates

$$\{[(\overline{y} = \overline{x} \wedge y_i = x_i \wedge Next\ \overline{y} = \overline{x'}) \rightarrow Next\ Afu(\overline{y} = \overline{x} \rightarrow [y_i = x_i \wedge Next\ \overline{y} = \overline{x'}])] :$$

$$i \in I \text{ and } \overline{x}, \overline{x'} \text{ and } \{x_i\} \text{ are disjoint}\}$$

where $\{y_i : i \in I\}$ is the set of all flexible symbols in our language. For our purposes we may assume that I is finite and then $Cw(\gamma)$ becomes a single formula. If for some reason it would be important to keep I infinite then $Cw(\gamma)$ is a set of formulas and everything goes through still.

Existence of weak clocks is a much weaker assumption than that of clocks, e.g. $Cw(\overline{y} = \overline{x})$ does not force the data domain to be infinite while $C(\gamma)$ does. Also $C(\overline{y} = \overline{x}) \vdash_{SF} Cw(\overline{y} = \overline{x})$. The results below generalize to the case when $\overline{y} = \overline{x}$ is replaced with arbitrary γ in Cw, but then Cw becomes longer to formulate (intuitive meaning remains the same).

THEOREM 12. $\vdash_{SFP} \equiv_{\square} (Ind + Tord)$ *generalizes to* **nondeterministic** *and concurrent programs under assuming existence of a weak clock (same for ordinary clocks).* ∎

§7. Temporal formulas in general

THEOREM 13. *Even if we permit flexible predicate- and function symbols, the temporal logics* $\vdash_{SF}$ *and* $\vdash_{SFP}$ *are complete for the semantics* $(Ind + Tord)$ *under assuming the existence of a clock. I.e. for any temporal formulas* φ *and* γ, $(i) \Longleftrightarrow (ii)$ *below.*

$$\begin{array}{lrcl} (i) & Ind + Tord + C(\gamma) & \models & \varphi \\ (ii) & C(\gamma) & \vdash_{SFP} & \varphi\ . \end{array}$$

The same holds with everything restricted to the language of $\vdash_{SF}$. ∎

The above theorem generalizes the completeness result for "T_1" in ABADI [1], [2] to HILBERT-style proof systems. In particular, for the formula φ constructed in §§3–4 therein to show incompleteness w.r.t. $(Ind + Tord)$ of the HILBERT-style T_0 found therein, we have $\vdash_{SF} \varphi_{Abadi}$ (while $\nvdash_{T_0} \varphi_{Abadi}$). The following improve the above result by weakening the clock assumption.

THEOREM 14. $\vdash_{SFP}$ *is complete for* $(Ind + Tord)$ *under assuming weak clocks. I.e. for any temporal* φ, $(i) \Longleftrightarrow (ii)$ *below.*

$$\begin{array}{lrcl} (i) & Ind + Tord + Cw(\overline{y} = \overline{x}) & \models & \varphi \\ (ii) & Cw(\overline{y} = \overline{x}) & \vdash_{SFP} & \varphi \end{array}$$

Note that the choices of φ *and* $\overline{y}$ *are independent.* ∎

THEOREM 15. *Theorem 14 fails for* $\vdash_{SF}$ *in place of* $\vdash_{SFP}$, *i.e.* $\vdash_{SF}$ *is incomplete w.r.t.* $(Ind + Tord)$ *under* $Cw(\overline{y} = \overline{x})$ *(but not under* $C(\gamma)$ *by Theorem 13).* ∎

THEOREM 16. *$\vdash_0$ is complete for the semantics Ind_{qf} ; i.e. for any temporal formula φ in the language of $\vdash_0$,*

$$\vdash_0 \quad \text{iff} \quad Ind_{qf} \models \varphi \,. \quad \blacksquare$$

In connection with the problems at the end of ABADI [1], [2] we obtain the following, even if we allow only constants as flexible symbols:

(17). *The inference system introduced in* MANNA–PNUELI [18] *for temporal logic with Next and Afu is incomplete w.r.t. the semantics $(Ind + Tord)$. Moreover, it remains incomplete after adding all propositionally valid formulas (cf. (A1) herein) and* **any** *finite number of* **new axioms** *(valid in $(Ind + Tord)$).*

(18). *The inference systems T_0, T_1 introduced in* ABADI*'s papers [1], [2] are incomplete w.r.t. $(Ind+Tord)$ if used without "Until". This is so even under assuming weak clocks and despite of T_1's being reinforced with somewhat unusual rules permitting the use of auxiliary definitions in proofs. By the remark at the end of §7 of the full version of* [1] *discussing "T_1 without Until", we conclude that clocks do increase the power of this system.*

Problem 19. *Are Theorems 12 or 14 true without assuming any kind of clocks?* ∎

Problem 20. *Is $\vdash_S \equiv_\Box Ind_1$ true for nondeterministic programs, where $Ind_1 (\subseteq Ind)$ is induction over NLP formulas containing at most one variable of sort T ?* ∎

REFERENCES

1. M.Abadi, *The power of temporal proofs*, Proceedings of the Second Annual IEEE Symposium on Logic in Computer Science, Ithaca, NY, USA; (1987), 123-130, Full version of this is: The power of temporal proofs, preprint of Digital Systems Research Center, 1988.
2. M.Abadi, *Temporal logic was incomplete only temporarily*, Preprint (1989).
3. M.Abadi and Z.Manna, *A timely resolution*, First Annual Symposium on Logic in Computer Science (1986), 176–189.
4. H.Andréka, K.Balogh, K.Lábadi, I.Németi, P.Tóth, *Plans to improve our program verifier program (in Hungarian)*, Working Paper, NIM IGÜSZI, Dept. of Software Techniques, Budapest (1974).
5. H.Andréka, I.Németi, and I.Sain, *A complete logic for reasoning about programs via nonstandard model theory, Parts I–II*, Theoretical Computer Science Vol **17** Nos **2, 3** (1982), 193–212 and 259–278.
6. H.Andréka, I.Németi, and I.Sain, *Temporal logics of programs with "binary" modalities*, Extended abstract (1989).
7. J.Barwise (ed), *Handbook of Mathematical Logic*, North–Holland (1977).
8. B.Biró and I.Sain, *Peano Arithmetic for the Time Scale of Nonstandard Models for Logics of Programs*, Annals of Pure and Applied Logic, to appear.
9. L.Csirmaz, *Programs and program verification in a general setting*, Theoretical Computer Science Vol **16** (1981), 199-210.
10. D.Gabbay and F.Guenther (eds), *Handbook of philosophical logic*, D.Reidel Publ. Co. Vol **II** (1984).
11. T.Gergely and L.Úry, *First order programming theories*, SZÁMALK Technical Report Budapest (1989), 232pp.
12. D.Gabbay, A.Pnueli, S.Shelah, J.Stavi, *On the temporal analysis of fairness*, Preprint Weizman Institute of Science, Dept. of Applied Math. (1981).
13. R.Goldblatt, *Logics of time and computation*, Center for the Study of Language and Information, Lecture Notes Number **7** (1987).

14. P.Hájek, *Some conservativeness results for nonstandard dynamic logic*, In: Algebra, combinatorics, and logic in computer science, Proc. Conf. Győr Hungary 1983 (eds: J.Demetrovics, G.Katona, A.Salomaa), Colloq. Math. Soc. J. Bolyai Vol **42**, North–Holland (1986), 443–449.
15. D.Harel, *First order dynamic logic*, Springer Lecture Notes in Computer Science Vol **68** (1979).
16. L.Henkin, J.D.Monk, and A.Tarski, *Cylindric Algebras Part II*, North–Holland (1985).
17. O.Lichtenstein, A.Pnueli, and L.Zuck, *The glory of the past*, Proc. Coll. Logics of Programs, Brooklyn, USA, Springer Lecture Notes in Comp. Sci. (ed: R. Parikh) Vol **193** (1985), 196–218.
18. Z.Manna and A.Pnueli, *The modal logic of programs*, International Colloquium on Automata, Languages and Programming'79, Graz, Springer Lecture Notes in Computer Science Vol **71** (1979), 385-409.
19. Z.Manna and A. Pnueli, *A hierarchy of temporal properties*, preprint (1986).
20. Z.Manna and R.Waldinger, *Is "sometime" sometimes better than "always"?*, Comm. ACM Vol **21** (1978), 159–172.
21. R.Parikh, *A decidability result for second order process logic*, IEEE Symposium on Foundation of Computer Science (1978), 177–183.
22. A.Pnueli, *Specification and development of reactive systems*, Information Processing (IFIP'86), H.–J. Kugler (ed.) North–Holland Vol **86** (1986), 845–858.
23. I.Sain, *There are general rules for specifying semantics: Observations on Abstract Model Theory*, CL and CL (Computational Linguistics and Computer Languages) Vol **XIII** (1979), 195–250.
24. I.Sain, *Structured Nonstandard Dynamic Logic*, Zeitschrift für Math. Logic u. Grundlagen der Math. Heft **3**, Band **30** (1984), 481–497.
25. I.Sain, *Nonstandard Logics of Programs*, Dissertation, Hungarian Academy od Sciences, Budapest (in Hungarian) (1986).
26. I.Sain, *Total correctness in nonstandard logics of programs*, Theoretical Computer Science Vol **50** (1987), 285–321.
27. I.Sain, *Elementary proof for some semantic characterizations of nondeterministic Floyd–Hoare logic*, Notre Dame Journal of Formal Logic, to appear (1989).