ON THE COMPLETENESS OF PROGRAM VERIFICATION METHODS

This is an abstract of a paper to be submitted to Acta Cybernetica, Szeged.

We use the same notions and notations as [1] especially we use the 3-sorted time-models, the traces, the Naur-Floyd-Hoare inference rules. In the appendix we summarize the notations and notions we use without references. We use $Ax_o$ a slight modification of the axiom system $Ax_o$ used in [1]. The most important modification is the requirement of the axiom of induction for every 3-sorted formula, i.e. for each 3-sorted formula $\varphi(z) \in F_{td}$ we require the validity of the following formula $I_{\varphi}$ where $z$ is a variable of sort time $t$:

$$I_{\varphi} = (\varphi(O) & (\forall z)(\varphi(z) \rightarrow \varphi(Sz)) \rightarrow (\forall z)\varphi(z))$$

Here $S$ is the unary operation symbol “successor”. In the forthcoming we always suppose that our models satisfy $Ax$.

**Definition 1.** Let $p = \langle (i_0 : u_0), (i_1 : u_1), \ldots, (i_n : HALT) \rangle \in P_d$ be a $d$-type program. $Th \subseteq F_{td}$ a $d$-type theory, then we say that $p$ is $Th$-allowed iff $Th \models i_j \neq i_k$ for every $j \neq k \leq n$. Let the set of $Th$-allowed $d$-type programs be $P_d(Th)$.

If $Th$ is the complete theory of a model $M$ then we write $P_d(M)$ rather than $P_d(Th(M))$.

**Definition 2.** Let $p = \langle (i_0 : u_0), (i_1 : u_1), \ldots, (i_n : HALT) \rangle$ be a program of type $d$. $M = \langle T, D, I, ext \rangle \in M_{td}, s \in I^{\omega+1}$ a trace of $p$ in $M$. Recall that $s(\omega)$ is the intension of the control variable. We say that $p$ terminates in $M$ iff there exists a $z \in T$ such that $ext(s(\omega), z) = i_n$. We say that $p$
ω-terminates in $M$ iff there exists a standard element $z = SS\ldots SO \in T$ such that $\text{ext}(s(\omega), z) = i_n$.

**Theorem 1.** Let $Ax \subseteq Th \subseteq F_{td}$, $p \in p_d(Th)$. If $p$ $\omega$-terminates in every model of $Th$ then $p$ is equivalent to a loop-free program (modulo $Th$).

This theorem shows that it can not be proved in first order way that even a slightly complicated program terminates within finite time.

**Proposition 2.** There exists a consistent theory $Th \subseteq F_{td}$ and a program $p \in p_d(Th)$ such that $Ax \subseteq Th$ and $p$ terminates in every model of $Th$ but $p$ is not equivalent to any loop-free program (modulo $Th$).

**Definition 3.** Call a time-model $M = \langle T, D, I, \text{ext} \rangle$ standard iff $T = \langle \omega; O(i + 1 : i \in \omega) \rangle = d_\omega I \subseteq D^2$ and $\text{ext}(f, a) = f(a)$.

Call $M$ quasi-standard if it fulfils the above assumptions but instead of the assumption of $T$ being $\omega$ we require the existence of a ZFC-model $W$ such that $T$ is isomorphic to the “set of natural numbers equipped with the zero and successor function” defined in $w$.

**Definition 4.** Let $M \models (p, \psi)$ iff whenever $s$ is a trace of $p$ in $M$

$$M \models (\forall z)(\text{ext}(s(\omega), z) = i_n \rightarrow \phi(\text{ext}(s(\omega)z, \ldots, \text{ext}(s(m), z)))$$

Let $Th \subseteq F_{td}$. We say that $Th \models (p, \psi)$ iff for every time-model $M$ satisfying $ThM \models (p, \psi)$, we say that $Th \models (p, \psi)$ for every standard time-model $M$ satisfying $ThM \models (p, \psi)$; we say that $Th \models (p, \psi)$ for every quasi-standard and standard time-model $M$ satisfying $ThM \models (p, \psi)$; finally we say that $Th \models^F (p, \psi)$ iff there exists a Naur-Floyd-Hoare proof of $(p, \phi)$ from $Th$. Let $d$ be an expansion of $t_2$, where $t_2$ is the similarity type of the arithmetic i.e. $t_2 = \{\langle 0, 0 \rangle, \langle s, 1 \rangle, \langle +, 2 \rangle, \langle \cdot, 2 \rangle\}$. Let $PA_d$ be the Peano-axiom system for similarity type $t_2$, let $PA_d = PA_2 \cup \{I^*_\phi : \phi(x) \in F_d\}$ where $I^*_\phi = ((\phi(0) \land (\forall x)(\phi(x) \rightarrow \phi(Sx))) \rightarrow (\forall x)(\phi(x))$ (the $d$-type induction axiom for $\phi$).

The following incompleteness theorem is well-known (see [3]).

**Theorem 3.** If $d$ is an expansion of $t_2, Ax \subseteq PA_d \subseteq Th \subseteq F_{td}$, then $\{(p, \phi) : p \in F_d(Th), Th^\omega \models (p, \phi)\}$ is not recursively enumerable.
However the following completeness theorem is valid:

**Theorem 4.** Let $d$ be a countable similarity type which is an expansion of $t_2$, $\Sigma \subseteq F_d$, $Th = \Sigma \cup Ax \cup PA_d \subseteq (F_d, p \in P_d(Th) \phi \in F_d$ then the following (i) – (iii) are equivalent:

(i) $Th \models (p, \psi)$  
(ii) $Th \models^q (p, \psi)$  
(iii) $Th \vdash^F (p, \psi)$

The following proposition shows that the $PA \subseteq Th$ condition is not superfluous in the above theorem:

**Proposition 5.** There exists a $Th \subseteq F_{td}$, a $p \in P_d(Th)$, $\psi \in F_d$ such that $Ax \subseteq Th$, $Th \models (p, \psi)$ but $Th \vdash^F (p, \psi)$ is not valid true.

**Theorem 6.** Let $Ax \subseteq Th \subseteq F_{td}$, $Th$ be recursively enumerable, then $\{(p, \phi) \in P_d(Th)xF_d : Th \models (p, \psi)\}$ and $\{p \in P_d(Th) : p$ terminates in every model of $Th\}$ is recursively enumerable.

Appendix: Notions and notations

$\omega$ is the set of natural numbers  
$\omega + 1 = \omega \cup \{\omega\}$

$d$ always denotes a classical similarity type  
$t$ denotes the similarity type of the Peano-arithmetic without the addition and multiplication, i.e. $t = \{(o, o), (s, l)\}$, where $s$ is the symbol of the successor function  
$t_2$ denotes the similarity type of the full Peano-arithmetic, i.e. $t_2 = \{(o, o), (s, l), (+, 2), (\cdot, 2)\}$

Let $d$ be any classical (or many-sorted) similarity type then  
$F_d$ denotes the set of $d$-type formulas,  
$M_d$ denotes the class of $d$-type models.
If $D$ is any classical model let $D$ be its underlying set.

Now we define

$P_d$ The set of programs of type $d$. We use the following symbols:

$Lab$ The set of label symbols is defined to be the set of all nullary terms of type $d$.

Logical symbols: \{&, ∀, =, ¬\}

Other symbols: ≺−, IF, GO TO, HALT; :, (, )

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

$(i : x_j \leftarrow \tau) \in U_d$ if $i \in Lab$, $x_j$ is a variable symbol of type $d$, $\tau$ is a $d$-type term, and $j \in \omega$

$(i : IF \chi GO TO v) \in U_d$ if $i, v \in Lab$; $x \in F_d$ is an open formula.

$(i : HALT) \in U_d$ if $i \in Lab$ these are the only elements of $U_d$.

By a program of type $d$ we mean a finite sequence $p$ of elements of $U_d$ with properties (i) – (iv) as follows:

(i) $p$ ends with a “HALT” command.
(ii) no two different members of $p$ have the same label
(iii) $p$ contains only one “HALT” command
(iv) if $p$ contains a command “IF $\chi$ GO TO $v$” then it contains a command labeled with $v$ as well

$P_d$ is the set of $d$-type programs.

For $P_d(Th)$ and $P_d(M)$ see Definition 1.

For each similarity type $d$ define the class of 3-sorted time-models.

$M_{td}$ as follows (for many-sorted languages see e.g. [2])

A typical element of $M_{td}$ is $M = (T, D, I, ext)$ where $T \in M_t$, $D \in M_d$, $I \in M_i$, $ext : I \times T \rightarrow D$.

The three sorts are $t$ (time), $d$ (data) and $i$ (intensions).
$F_{td}$ is the usual 3-sorted first order language of the time-models. The
variables of the sort $t, d$ and $i$ are $z_j, x_j$ and $y_j$ respectively.

(j $\in \omega$)

Let $M = (T, D, I, ext) \in M_{td}$, $p = ((i_0, u_0), \ldots, (i_n : HALT)) \in P_d(M)$
s $\in I^{\omega+1}$ then we say that $s$ is a trace of $p$ in $M$ if the following (i) – (iii)
holds:

(i) $ext(s(\omega), 0) = i_0$

(ii) if $ext(s(\omega), z) = i_n$ then for every $j \leq \omega$ $ext(s(j), sz) = ext(s(j), z)$

(iii) if $ext(s(\omega), z) = i_m (m < n)$ then

a) if $u_m = (x_i \leftarrow \tau(x_0, x_1, \ldots, x_k))$ then
$ext(s(j), sz) = ext(s(j), z)$ (i $\neq j \in \omega + 1$ and $ext(s(i), sz)$
$= \tau(ext(s(o), z), \ldots, ext(s(k), z))$

b) if $u_m = (IF \chi(x_0, \ldots, x_k) \text{GO TO } v)$ then $ext(s(j), sz) = ext(s(j), z)$ for
$j \in \omega$ and if $D \models \chi(ext(s(0), z), \ldots, ext(s(k), z))$ then $ext(s(\omega), sz) = v$;
in the other case $ext(s(\omega), sz) = i_{m+1}$

$Ax \subseteq F_{td}$ is defined as the union of $PA$ and the set of induction axioms, i.e.
$Ax = \{sz_1 = sz_2 \rightarrow z_1 = z_2, sz_1 \neq 0\} \cup \{I_\varphi : \varphi(z) \in F_{td}\}$. (Here $PA$
means $PA_2 \cap F_{I_1}$).

Let $Ax \subseteq Th \subseteq F_{td}$, $p = ((i_0 : u_0), \ldots, (i_n : HALT)) \in P_d(Th)$, $\phi \in F_d$.
Let $lab(p) = \{i_m : m \leq n\}$. A Floyd-proof of $(p, \phi)$ from $Th$ is a mapping
$\phi : lab(p) \rightarrow F_d$ equipped with classical proofs (i) – (iv)

(i) a proof $Th \vdash \phi(i_0)$

(ii) For every command $i_m : x_w \leftarrow \tau$ a proof $Th \vdash \phi(i_m)$ & $\tau = x_w \rightarrow$
$(\exists x_w)(x_w = x_\omega \& \phi(i_{m+1}))$.

(iii) For every command $i_m : IF \chi \text{ GO TO } v$ a proof $Th \vdash (x \& \phi(i_m) \rightarrow$
$\phi(v))$ and a proof $Th \vdash (\neg x \& \phi(i_m) \rightarrow \phi(i_{m+1}))$

(iv) a proof $Th \vdash \phi(i_n) \rightarrow \phi$

$Th \vdash^F (p, \phi)$ denotes that there exists a Floyd proof of $(p, \phi)$ from $Th$. 
References


Institute for Coordination of Computer Techniques
H–1308 Budapest, Pf. 224
Hungary