Bulletin of the **S**ection of **L**ogic
Volume 8/3 (1979), pp. 124–128

reedition 2010 [original edition, pp. 124–129]

Hajnal Andréka
István Németi
Ildikó Sain

PROGRAM VERIFICATION WITHIN AND WITHOUT LOGIC

Abstract

Theorem 1 states a negative result about the classical semantics $\models^{\omega}$ of program schemes. Theorem 2 investigates the reason for this. We conclude that Theorem 2 justifies the Henkin-type semantics $\models$ for which the opposite of the present Theorem 1 was proved in [1]–[3] and also in a different form in part III of [5]. The strongest positive result on $\models$ is Corollary 6 in [3].

Basic concepts

First we recall some basic notions and notations from textbooks on Logic [7], [4] and from Program Scheme Theory e.g. [6], [1]–[3], [5].

ω denotes the set of natural numbers.
d denotes an arbitrary similarity type. I.e.: d correlates arities to some fixed function symbols and relation symbols.
$Y = \{y_z : z \in \omega\}$ denotes the set of variable symbols.
F_d is the set of all classical first order formulas of type d with variables in Y.
M_d is the class of all classical first order *models* of type d.
$\models \subseteq F_d \times M_d$ is the usual validity relation.
τ denotes a *term* of type d in the usual sense of first order logic, see [4], p. 22 or [7], p. 166 D.10.8.(ii).

$\underline{D}$ and $\underline{E}$ denote elements of M_d the universes of which are D and E respectively.
P_d denotes the set of *program schemes* of type d. P_d is defined as in [6], [1], [2], [5], p. 72. E.g. let t be the similarity type of arithmetic. Then the following sequence is an P_t, i.e. it is a program scheme of type t:

$< (0 : y_0 \leftarrow\!\!\!\leftarrow 0),$
$(1 : \text{IF } y_0 = y_1 \text{ THEN } 1 + 1 + 1 + 1),$
$(1 + 1 : y_0 \leftarrow\!\!\!\leftarrow y_0 + 1),$
$(1 + 1 + 1 : \text{IF } y_1 = y_1 \text{ THEN } 1),$
$(1 + 1 + 1 + 1 : \text{HALT}) >.$

$P_d \times F_d$ is the set of *output statements* about programs. An output statement $(p, \phi) \in P_d \times F_d$ means intuitively that be program scheme p is partially correct w.r.t. output condition ϕ.
$\underline{D} \models^{\omega} (p, \phi)$ is meaningful if $\underline{D} \in M_d$ and $(p, \phi) \in P_d \times F_d$. Now, $\underline{D} \models^{\omega} (p, \phi)$ holds if the program scheme p is partially correct w.r.t ϕ in the model $\underline{D}$. I.e. if p is started in $\underline{D}$ with any *input* $q : \omega \to D$ then whenever p *halts* with some output $k : \omega \to D$, the formula ϕ will be true in $\underline{D}$ under the valuation k of its free variables, i.e. $\underline{D} \models \phi[k]$. See [6] Chapter 4.

Note that a precise definition of $\models^{\omega}$ would strongly use the structure $\langle \omega, \leqslant \rangle$ of *natural numbers*. See [5], p. 78, [1], p. 116, [2], [3]. The letter ω above the sign $\models$ serves to remind us of this fact.

For any set $Th \subseteq F_d$ of formulas, "$Th \models^{\omega} (p, \phi)$" is defined in the usual way: $(\forall \underline{D} \in M_d)[\underline{D} \models Th \Rightarrow \underline{D} \models^{\omega} (p, \phi)]$.

From now on c and τ denote arbitrary terms of type d such that c contains no variables and τ contains one variable y_0. To make this explicit we write $\tau(y_0)$.
Notation: $\tau^0 =^{df} c$, and $\tau^{z+1} =^{df} \tau(\tau^z)$ for every $z \in \omega$. Note that the terms $\tau^0, \tau^1, \ldots, \tau^z, \ldots$ contains no variables.

Definition. $Th \subseteq F_d$ is *good* if there exist terms c and $\tau(y_0)$ such that $Th \supseteq \{\tau^z \neq \tau^r : z < r \in \omega\} =^{df} Th'$. Let $\underline{E} \in M_d$ be an arbitrary model of Th' such that $(\forall b \in E)(\exists z \in \omega)[\tau^z$ in E denotes $b]$. Then

$May(Th) =^{df} \{(p, \phi) \in P_d \times F_d : Th \models^{\omega} (p, \phi)\}$.

$Must(Th) =^{df} \{(p, \phi) \in May(Th) : p$ terminates in $\underline{E}$ for every input, and ϕ is an *atomic formula* or the negation of it and $Th \models y_0\phi\}$.

Remark. To a fixed Th, $Must(Th)$ is not unique since it may depend on the choice of $c, \tau(y_0)$ and $\underline{E}$. This makes the following theorem even stronger since it will hold for any choice of c, τ and $\underline{E}$. Observe that $Must(Th)$ is a reasonably small set of output statements since ϕ contains no quantifiers, no "$\vee$" or "$\wedge$" and at the same time p is such that it terminates in $\underline{E}$ for every input. (Thus $Must(Th)$ contains no tricky statement about the "halting problem" (since p has to terminate) and no "strange sentence" since ϕ has to be simple).

Theorem 1. *Let d be arbitrary and let $Th \subseteq F_d$ be good and consistent. Let H be an arbitrary set such that $May(Th) \supseteq H \supseteq Must(Th)$. Then H is not recursively enumerable.*

The following theorem says that if one "avoids Logic" and proves properties of programs by using "Mathematics in general" then this will *not help* one to avoid the "shortcoming" formulated in Theorem 1.

Theorem 2. *Let the real world $\langle V, \in\rangle \models ZFC$ of set theory (see [9], p. 3 or [4], p. 476) be fixed. I.e. V is the class of all sets and $\in$ is the "element of" the relation between then.*

Then there exist

- *a similarity type d, and*
- *a model $\langle W, E\rangle \models ZFC$ of set theory inside of $\langle V, \in\rangle$ (i.e. $\langle W, E\rangle$ is an element of V and $\langle W, E\rangle \models ZFC$ is true inside $\langle V, \in\rangle$, see [9], p. 14) such (i) and (ii) below hold.*

(i) There is a finite set $Th \subseteq F_d$ of axioms and an output statement (p, ϕ) such that
$Th \models^\omega (p, \phi)$ is true, but inside $\langle W, E\rangle$ we have $Th \not\models^\omega (p, \phi)$.
More precisely:
$\langle V, \in\rangle \models$ "$Th \models^\omega (p, \phi)$" but
$\langle W, E\rangle \models$ "$Th \not\models^\omega (p, \phi)$".
(Observe that "$Th \models^\omega (p, \phi)$" is a statement on the language of ZFC).

(ii) There is an output statement (p, ϕ) such that
$\langle V, \in\rangle \models$ "$M_d \models^\omega (p, \phi)$" while
$\langle W, E\rangle \models$ "$M_d \not\models^\omega (p, \phi)$".

As a *contrast* we note that: For all $\phi \in F_d$ and for every model $\langle W, E\rangle \in V$ of ZFC,

$$\langle V, \in\rangle \models \text{“}M_d \models \phi\text{” implies } \langle W, E\rangle \models \text{“}M_d \models \phi\text{”}.$$

The above Theorem 2 says that something is wrong with the classical semantics (or model theory $\models^\omega$ of program schemes. Namely: there exists a good program (p, ϕ) which is not provable from ZFC despite of the fact that (p, ϕ) *is* good. See [8] D.2 about “$Th \models^\omega (p, \phi)$”-s being a formula of Set Theory. In this way Theorem 2 supports the Henkin-type semantics introduced in [1]–[3] which is well presented and the consequence concept $(Th \models (p, \phi))$ of which does not have the above shortcoming.

References

[1] H. Anfréka and I. Németi, *Completeness of Floyd Logic*, **Bulletin of the Section of Logic**, Vol. 7 (1978), No 3, pp. 115–120.

[2] H. Andréka and I. Németi, *A characterisation of Floyd provable programs*, preprint 1978/8 of Math. Inst. H. A. S. Abstracted in **Bulletin of the Section of Logic**, Vol. 7 (1978), pp. 115–120.

[3] H. Andréka and I. Németi, *Classical many-sorted model theory to turn negative results on program schemes to positive*, preprint Math. Inst. H. A. S. 1979.

[4] C. C. Chang and H. J. Keisler, **Model theory**, North Holland 1973.

[5] T. Gergely and L. Ury, **Mathematical Programming Theories**, Preprint 1979. SZAMKI Budapest, Csalogány 30, H-1536.

[6] Z. Manna, **Mathematical Theory of Computation**, McGraw Hill 1974.

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

[8] I. Németi and I. Sain, *Connections between Algebraic Logic and Initial Algebra Semantics of CF languages*, submitted to **Proc. Coll. Logic in Programming Salgótarján** 1978.

[9] K. J. Devlin, **Aspects of Constructibility. Lecture Notes in Mathematics**, 354, Springer Verlag 1973.

Mathematical Institute of the
Hungarian Academy of Sciences
Budapest, Reáltanoda u. 13–15
H–1053 Hungary