ON RELATIVE COMPLETENESS OF PROGRAMMING LOGICS

(Extended abstract)

Michal Grabowski
Institute of Informatics
University of Warsaw
PkiN p.o. Box 1210
00-901 Warsaw POLAND

Abstract In this paper a generalization of a certain Lipton's theorem (see Lipton [5]) is presented. Namely,we show that for a wide class of programming languages the following holds:the set of all partial correctness assertions true in an expressive interpretation I is uniformly decidable (in I) in the theory of I iff the halting problem is decidable for finite interpretations. In the effect we show that such limitations as effectiveness or Herbrand definability of interpretation (they are relevant in the previous proofs) can be removed in the case of partial correctness.

1.BACKGROUND

In this section we recall some history of the considered problem and we restate the known results.

In order to show the inherent complexity of the problem of partial correctness Cook introduced the notion of relative completeness. Supplying Hoare's system with an oracle answering questions on validity of first-order formulas he was able to separate the reasoning about the programs from the reasoning about the undrelying language of ivariants. The idea of oracle results in Hoare-like system for programming language which is considered in [3]. This system is relatively complete, i.e. complete over expressive interpretations. (An interpretation I is said to be expressive iff the weakest preconditions of programs are first-order definable in I.)

Natural question arose for other,more complicated programming languages : does the expressiveness stand for the sufficient condition for the existence of relatively complete Hoare's logic ?

Clarke (see [1]) discovered that for languages with certain natural features (e.g. call by name parameter passing,functions,global variables and coroutines with local recursive procedures that can access global variables) it is impossibl e to construct a Hoare's logic which is sound and relatively complete in the sense of Cook.This incompleteness result is based on the observation that if a programming language possesses a relatively complete proof system for partial correctness assertions then the halting problem for finite interpretations must be decidable.

Lipton (see [5]) attempted to prove the convers e : if PL is an acceptable programming language and the halting problem for programs in PL is decidable for finite interpretations then PL has a relatively complete proof system for partial correctness assertions.

Roughly speaking, a programming language PL is said to be acceptable iff every program in PL can be effectively translated into, for instance, Friedman's scheme (see [4]) and PL is closed under reasonable programming constructs. Eventually, Lipton obtained the following partial answer :

Th.1 (Lipton, 1977)

Let PL be a deterministic acceptable programming language. Then the following are equivalent:

1. PL has a decidable halting problem for finite interpretations.

2. The true quantifier-free partial correctness assertions are recursively enumerable in Th(I) and in a certain presentation of I for expressive and effective interpretations I.

Clarke, German and Halpern (see [2]) obtained a significant generalization of the Lipton's theorem to the first-order partial (and total too) correctness assertions. Their results are quoted below.

An interpretation I is said to be Herbrand-definable iff every element of I is the value of a constant term.

Th.2 (Clarke, German, Halpern 1982)

Let PL be a deterministic acceptable programming language with recursion. Then the following are equivalent :

1. PL has a decidable halting problem for finite interpretations.

2. The true first-order partial (resp. total) correctness assertions are uniformly (in I) decidable in Th(I) for expressive and Herbrand-definable interpretations I.

If we limit ourselves to expressive interpretations, then the following holds :

Th.3 (Clarke, German, Halpern 1982)

Let PL be a deterministic acceptable programming language. Then the following are equivalent:

1. PL has a decidable halting problem for finite interpretations.

2. The true first-order partial (resp. total) correctness assertions are decidable in Th(I) and in a certain presentation of I for expressive and effective interpretations I.

Notice that a decision procedure (in Th.3) for correctness assertions depends simultaneously on Th(I) and on I, i.e. it is not uniform in I. It means that such a procedure does not stand for a realistic analogue of Hoare-type proof system, since Hoare-type proof systems are independent of the particular concrete interpretation.

2. RELATIVE COMPLETENESS OF PARTIAL CORRECTNESS

In this section our main result is presented. We shall prove the following

Th.4

Let PL be a deterministic acceptable programming language with recursion. Then the following are equivalent :

1. PL has a decidable halting problem for finite interpretations.

2. The true first-order partial correctness assertions are uniformly decidable in Th(I) for expressive interpretations I.

It seems that theorem 4 (comparing to th.2, th.3) provides more information on ability to find good axiom system for complicated programming languages In words of Clarke, german and Halpern [2]:

"In order for a decision procedure to be a realistic analogue of a Floyd-Hoare axiom system it should, in some sense, be uniform; i.e. independent of the particular interpretation that is being used."

In order to outline the proof of our result some definitions and notions are necessary. The basic one is the notion of an acceptable programming language. We do not quote the long definition and we refer the reader to the paper [2]. Intuitively, a programming language PL is said to be acceptable iff for every program in PL it is possible to effectively ascertain its step-by-step computatio n in interpretation I by checking in I open formulas; moreover, PL is closed under reasonable programming constructs. For instance, almost all

Algol-like programming languages are acceptable.

Let PL be an acceptable programming language.

For a program P in PL and for an interpretation I , $A_P^I(x,y)$ denotes the input-output relation of the program P in the interpretarion I.

def. An interpretation I is expressive (for PL) iff for every program P in PL there is a first-order formula $B_P(x)$ such that I $B_P(x)$ iff for some y in I, $A_P^I(x,y)$.

def. $I \models \varphi\{P\}\psi$ iff for all x,y in I, if $I \models \varphi(x)$ and $A_P^I(x,y)$ then $I \models \psi(y)$.

def. An interpretation I is weakly arithmetic iff there exist first-order formulas N(x), E(x,y), S(x,y), Z(x), Add(x,y,z), Mult(x,y,z) (with respectively k, 2k, 2k, k, 3k and 3k free variables for some k) such that E defines an equivalence relation on I^k and formulas N, Z, S, Add, Mult define on the set $\{x \mid I \models N(x)\}$ the model M such that the quoteient model M/E is isomorphic to the standard model $\langle \omega; 0, +1, +, *, = \rangle$.

Now we are in a position to outline the proof of theorem 4.

Let I be an expressive interpretation.

As in Lipton [5], our proof splits into two cases.

The case when for every program P in PL there is a number B such that P never accesses more than B values on any input was proved by Clarke, German and Halpern in [2].

In the case when some program can access an unbounded number of different values our approach is different from that of Clarke, German and Halpern. The key idea is to represent the input-output relation of a program by means of the least relations satisfying certain first-order conditions. We join fixed-point approach and ideas of coding of terms used by Clarke, German and Halpern in [2].

Let P be in PL and let $x=\{x_1,\dots,x_q\}$ be the set of free variables of the program P (dep(P) in [2]) Let $y = \{y_1,\dots,y_q\}$ be a copy of x. Let $\underline{N},\underline{E},\underline{Z},\underline{S},\underline{Add}$, $\underline{Mult}$ be new predicate symbols for arithmetical notions, $\underline{H},\underline{U},\underline{F}$ be (respectively) 2-ary, 1-ary and and 2-ary new predicate symbols.

<u>Lemma</u>

We can effectively construct first-order axioms Fnc (encoding) for $\underline{H},\underline{U}$, and axioms SyntP for $\underline{F}$ and a first-order formula $InOut_P(\underline{N},\dots,\underline{Mult},\underline{H},\underline{U},\underline{F})$ such that :

1. For every first-order formulas N,...,Mult which model in I axioms Ax for arithmetic (AX1-9 in [2]) and for every first-order formulas H,U,F which model in I axioms Fnc and SyntP, the following holds :

for all x,y in I, if $A_I^P(x,y)$ then

$$I \models InOut_P(N,\dots,F)(x,y).$$

2. There exist first-order formulas $N_\omega,\dots,F_\omega$ such that they model in I axioms Ax, Fnc, SyntP and for all x,y in I , $A_P^I(x,y)$ iff

$$\text{iff} \quad I \models InOut_P(N_\omega,\dots,F_\omega)(x,y).$$

Proof (Outline)

The set Fnc consists of recursive definitions for coding of terms over variables $x \cup y$ (something like H(z,d) in [2]) and recursive definitions for universal predicate $\underline{U}(z)$ for <u>open</u> formulas over variables $x \cup y$: $\underline{U}(z)$ iff the z-th open formula over $x \cup y$ is satisfied. (Recursive definitions for $\underline{H},\underline{F}$ involve $\underline{N},\underline{E},\dots,\underline{Mult}$ and recursive definitions for $\underline{U}$ involve $\underline{H},\underline{N},\underline{E},\dots,\underline{Mult}$.)

We can construct a recursively enumerable sequence $\beta_0(x,y), \beta_1(x,y), \dots$ of open formulas such that $A_I^P(x,y)$ iff $I \models \beta_0(x,y) \vee \beta_1(x,y) \vee \dots$.

Let f(n) = standard cod of $\beta_n(x,y)$.

The set SyntP consists of recursive definitions (for $\underline{F}$) representing f (treated as a relation). We define

$$InOut_P(\underline{N},\dots,\underline{F}) = (\exists w)(\exists v)(\underline{N}(w) \wedge \underline{N}(v) \wedge \underline{F}(w,v) \wedge \underline{U}(v)).$$

Let I be the interpretation that is being considered. Since I is expressive, the theorem of deMillo, Lipton, Snyder (see [5]) imply that I is weakly arithmetic. The relations which are defined in I by first-order formulas which make I weakly arithmetic are programmable in I (as relations). This fact is derivable from the proof of the theorem of de Millo, Lipton, Snyder. Recall that PL is assumed to be deterministic and it is closed under recursion. Recursive definitions in Fnc and in SyntP "work right" on standard natural numbers. Hence, point 2 of our lemma holds by expressivity of I. □

We prove the theorem by making use of the following fixed-point rule (FR) :

premisses: Ax(N,...,Mult), Fnc(N,...,Mult,H,U), SyntP(N,...,Mult),

$\varphi(x) \rightarrow (\forall y)(\mathrm{InOut}_p(N,\dots,F)(x,y) \rightarrow \rightarrow \psi(y))$

(for certain first-order formulas N,...,Mult,H,U,F)

conclusion: $\varphi\{P\}\psi$.

Let $I \models \varphi\{P\}\psi$. After constructing a formula $\mathrm{InOut}_p(\underline{N},\dots,\underline{F})$ and sets $\mathrm{Fnc}(\underline{N},\dots,\underline{H},\underline{U})$, $\mathrm{SyntP}(\underline{N},\dots,\underline{Mult})$, algorithms guesses the formulas N,...,F such that the premisses of the rule (FR) are true in I, then applies the rule (FR).

The lemma implies correctness of pur algorithm.

Thus we have proved that the set of all partial assertions true in I is uniformly recursively enumerable in Th(I).

It remains to be proved that the set of all true in I negations of partial correctness assertions is uniformly recursively enumerable in Th(I).

Notice that it is possible to assign effectively to a program P a recursively enumerable sequence $\beta_0(x,y), \beta_1(x,y), \dots$ of open formulas such that

$$A_P^I(x,y) \text{ iff } I \models \beta_0(x,y) \vee \beta_1(x,y) \vee \dots$$

The theorem follows from the following fact:

$$I \models \neg\varphi\{P\}\psi \text{ iff there exists n in } \omega \text{ such that } I \models (\exists x,y)(\varphi(x) \wedge \beta_n(x,y) \wedge \neg\psi(y)).$$ □

Concluding remarks

1. Our method can not be transferred to the case of total correctness. This is solved by Clarke, German and Halpern in [2] for case of Herbrand--definable and expressive interpretations.

2. We do not use coding of finite sequences.

3. It seems that the proof of the theorem 4 suggests a way for constructing a relatively complete proof system for complicated programming languages. Namely, such a system should contain the rule (FR) and it should employ relational variables in order to make possible to construct a formula $\mathrm{InOut}_p(\underline{N},\dots,\underline{F})$.

4. Theorem 4 does not stand for a definite improvement of the Liptons theorem. Is it possible to remove the assumption that PL is closed under recursion ? Moreover, the problem of relative completeness of dynamic logics based on acceptable programming languages remains open.

REFERENCES

[1] Clarke, E.M., Programming language constructs for which it is impossible to ob tain good Hoare axiom systems. JACM 26,1, January, 1979

[2] Clarke, E.M. and German, S.M. and Halpern, J.Y. On effective axiomatizations of Hoare Logics in 9th POPL symp., January 1982, 309-321

[3] Cook S.A., Soundness and completeness of axiom system for program verification. SIAM Journ. on Comp. 7,1, pp.70-90, Febr. 1978

[4] Friedman, H., Algorithmic procedures, generalized Turing algorithms and elementary recursion theory. Gandy and Yates (eds) Logic Colloquim

[5] Lipton, R.J., A necessary and sufficient conditions for existence of Hoare Logics. 18th IEEE Symp. FOCS, pp.1-6, October 1977