6 Proving Success and Absence of Runtime Errors

6.1 Success

In order to prove success of a program $P = (L, T, s, t)$ we have to ensure that no computation of P ends in a failing configuration $\langle l; \sigma \rangle,\ l \not\equiv t$. Whenever we reach a such a configuration $\langle l; \sigma \rangle,\ l \not\equiv t$, we should ensure to have an enabled transition available, i.e., that $(l, c \to f, l') \in T$ exists with $c(\sigma) = tt$. The set of states which allow taking at least one transion can be derived from the set of transition originating from a certain location. For location $l,\ l \not\equiv t$ we define

$$T_l \stackrel{def}{=} \{(l_i, c \to f, l_j) \in T | l_i \equiv l\}.$$

For t this set is empty by definition.

Next we characterize the set of states for which some transition is enable at location l by the predicate

$$\mathcal{C}_l \stackrel{def}{=} (\bigvee_{(l, c \to f, l') \in T_i} c) \vee false$$

This predicate is notated by a disjunction over all guards of the transitions leaving l. In case there is no transition, this predicate is set to $false$.

Definition 6.1 (Floyd's inductive assertion method applied to proving success) Given a transition diagram $P = (L, T, s, t)$, in order to verify that it is φ-successful with respect to a precondition φ we use Floyd's method for proving success of sequential programs as formulated below:

1. Find an assertion network $\mathcal{Q}$, show that it is inductive, and that $\models \varphi \to \mathcal{Q}_s$ holds.
2. Show for each $l \not\equiv t$, that $\models \mathcal{Q}_l \to \mathcal{C}_l$ holds.

□

Next we prove soundness and semantic completeness of the method.

Theorem 6.2 (Soundness)
Let $P = (L, T, s, t)$. If $\mathcal{Q}$ is an inductive assertion network for P, $\models \varphi \to \mathcal{Q}_s$ holds, and $\models \mathcal{Q}_l \to \mathcal{C}_l$ holds for all $l \not\equiv t$, then every finite computation of P is successful.

Proof
Consider an finite φ-computation

$$\eta : \langle l_0; \sigma_0 \rangle \longrightarrow \langle l_1; \sigma_1 \rangle \longrightarrow \ldots, \longrightarrow \langle l_n; \sigma_n \rangle$$

with $l_0 = s$, then inductiveness of $\mathcal{Q}$ and $\models \varphi \rightarrow \mathcal{Q}_s$ implies that $\mathcal{Q}$ is invariant, i.e., that $\models \mathcal{Q}_{l_i}(\sigma_i)$ holds for $0 \leq i \leq n$.

Now assume that $l_n \not\equiv t$, i.e., that η is a failing computation. In case that some transition $(l_n, c \rightarrow f, l_{n+1}) \in T_{l_n}$ exists such that $\models c(\sigma)$ holds we have a contradiction to the fact that η is a computation of P, as η could be extended by a computation step according to this transition. Thus no such transition is enabled and consequently, we have $\mathcal{C}_{l_n}(\sigma_n) = ff$. But as $\models \mathcal{Q}_l \rightarrow \mathcal{C}_l$ holds for all $l \not\equiv t$ and $l_n \not\equiv t$, we deduce that $\mathcal{Q}_{l_n}(\sigma_n)$ evaluates to ff. This contradicts the fact that $\models \mathcal{Q}_{l_i}(\sigma_i)$ holds for $0 \leq i \leq n$.

■

Theorem 6.3 ((Semantic) Completeness)
Let $P = (L, T, s, t)$. If P is φ-successful, then there exists an inductive assertion network $\mathcal{Q}$ such that $\models \varphi \rightarrow \mathcal{Q}_s$ holds and $\models \mathcal{Q}_l \rightarrow \mathcal{C}_l$ holds for all $l \in L,\ l \not\equiv t$.

Proof
We choose the same assertion network $\mathcal{Q}$ as in the semantic completeness proof of the partial correctness method (Theorem 4.4). It is inductive, and $\models \varphi \rightarrow \mathcal{Q}_s$ holds.

It remains to prove that $\models \mathcal{Q}_l \rightarrow \mathcal{C}_l$ holds for all $l \in L,\ l \not\equiv t$. Thus let $l \in L,\ l \not\equiv t$, and $\sigma \in \Sigma$ such that $\models \mathcal{Q}_l(\sigma)$ holds. By definition of $\mathcal{Q}$, there exists σ' such that $\models varphi(\sigma')$ holds and $\langle s; \sigma' \rangle \longrightarrow^* \langle l; \sigma \rangle$. Since $l \not\equiv t$ and P is φ-successful, this can only be a proper prefix of a φ-computation of P. In particular we have a configuration $\langle l'; \sigma'' \rangle$ following $\langle l; \sigma \rangle$ in this φ-computation. This is possible only if there exists a transition $(l, c \rightarrow f, l') \in T$ such that $\models c(\sigma)$ holds, and $f(\sigma) = \sigma''$. As $\models c \rightarrow \mathcal{C}_l$ holds by construction, we obtain $\models \mathcal{C}_l(\sigma)$, i.e. $\models \mathcal{Q}_l \rightarrow \mathcal{C}_l$ holds.

■

6.2 Absence of Runtime Errors

Finally, we extend our notion of transition diagrams (and systems) by allowing them to contain partial state transformations. In particular, we consider the case that state transformations f occurring in instructions $c \rightarrow f$, which label an edge of a transition diagram, are partial, such as, e.g., the state transformations associated with $x := 1/0$ and $x := 1/y$. Clearly, executing $x := 1/0$, and $x := 1/y$ for $y = 0$, is undesirable because this leads to an undefined result, and, therefore, to runtime errors.

Consequently, in addition to the four modalities of program correctness defined in Session 2, we introduce a fifth one, called *absence of runtime errors*, and discuss how to prove this modality for a given program (and given precondition). Observe, that as a consequence of it being possible to model runtime errors in our formalism, total correctness amounts to the *sum* of partial correctness, convergence, success, and absence of runtime errors. That is, in order to prove total correctness of a program P w.r.t. precondition φ and postcondition ψ, one has

to prove that P is partially correct w.r.t. precondition φ and postcondition ψ, is φ-convergent, and displays neither deadlocks nor runtime errors when started in φ.

Definition 6.4 (Absence of runtime errors) Given a transition diagram P and a precondition φ, whenever no φ-computation of P ever reaches a location in which an enabled transition is undefined, one speaks of *absence of* (or *freedom from*) *runtime errors of* P *in* φ. □

We present as method for proving φ-absence of runtime errors a modification of Floyd's inductive assertion method.

Obtaining a method for proving φ-absence of runtime errors is at first sight straightforward. Just modify the second clause of the definition of Floyd's method by also proving

$$\models \mathcal{Q}_l \wedge c \rightarrow \mathit{Def}(f),$$

where $\mathit{Def}(f)$ denotes a predicate expressing the domain of definition of f, and delete in its third clause the condition $\models \mathcal{Q}_t \rightarrow \psi$.

However, how does one evaluate the verification condition $\models \mathit{false} \rightarrow \mathit{true} \circ (x := 1/0)$? Here the difficulty is that $\mathit{true} \circ (x := 1/0)$ denotes a nowhere-defined function.

In general, the possibility of state transformations f being partial functions leads to the occurrence of partially-defined boolean functions, when applying Floyd's method, and this forces one to modify the definitions of the boolean connectives accordingly, because these occur in the formulation of verification conditions.

Since a boolean function $\varphi : \Sigma \rightarrow \mathit{Bool}$, with $\mathit{Bool} \stackrel{\text{def}}{=} \{tt, ff\}$, can now be partial – e.g., consider $\mathit{true} \circ (x := 1/0)$, one distinguishes between three cases: that $\varphi(\sigma) = tt$, $\varphi(\sigma) = ff$ and $\varphi(\sigma)$ is undefined, for $\sigma \in \Sigma$; the latter case we abbreviate by $\varphi(\sigma) = \bot$, with "$\bot$" pronounced as "bottom". That is, instead of dealing with *partial* functions to Bool, we model them using *total* functions to $\mathit{Bool}_\bot$, with $\mathit{Bool}_\bot \stackrel{\text{def}}{=} \{tt, ff, \bot\}$, where $\varphi(\sigma) = \bot$ in the total-function view of φ iff $\varphi(\sigma)$ is undefined in the partial-function view of φ.

The meanings of $\neg$, $\rightarrow$, $\wedge$ and $\vee$ in this total-function view are given by:

1. definition of "$\neg$":

$\neg$	
tt	ff
ff	tt
$\bot$	$\bot$

2. definition of "$\rightarrow$":

$\rightarrow$	tt	ff	$\bot$
tt	tt	ff	$\bot$
ff	tt	tt	tt
$\bot$	tt	$\bot$	$\bot$

3. definition of "$\wedge$":

$\wedge$	tt	ff	$\bot$
tt	tt	ff	$\bot$
ff	ff	ff	ff
$\bot$	$\bot$	ff	$\bot$

4. definition of "$\vee$":

$\vee$	tt	ff	$\bot$
tt	tt	tt	tt
ff	tt	ff	$\bot$
$\bot$	tt	$\bot$	$\bot$

The motivation for these definitions is twofold:

1. It defines $tt \vee \bot = \bot \vee tt = tt$. When generalised to existential quantifiers this lets such reasonable predicates as $\exists x(1/x = 1)$ denote tt.
2. By ordering $Bool_\bot$ as follows: $\bot \sqsubseteq tt$, $\bot \sqsubseteq ff$, $Bool_\bot$ becomes a so-called *complete partial order* (cpo). In this structure we want $\wedge$, $\vee$ and $\rightarrow$ to denote monotone functions in their arguments, where f is *monotone* if:

$$a \sqsubseteq b \Rightarrow f(a) \sqsubseteq f(b).$$

It can be checked that requirements (i) and (ii) above determine the meanings of these propositional connectives uniquely, and that they coincide with the ones given above them. Consult, e.g., [dRE98, Chapter 8], for more details on this topic.

Next we face the choice between partial and total inductive assertions. Here we choose them to be total $\{tt, ff\}$-valued functions φ, $\varphi : \Sigma \rightarrow Bool$. That is, the predicates $\mathcal{Q}_l$ used for reasoning about transition diagrams in the inductive assertion approach are *total $\{tt, ff\}$-valued boolean functions,* even in the presence of runtime errors during the execution of transition diagrams. Observe that this is consistent with the five modalities of program correctness which we want to prove. For example, partial correctness means that only in the case of termination has some predicate to hold; and, since in that case the resulting state is well-defined, that predicate can be chosen to be total and $\{tt, ff\}$-valued. Or, in the case of success, one has to prove absence of blocking when the execution is started in a given precondition, which must, therefore, yield a well-defined starting state, and, hence, can be chosen to be total and $\{tt, ff\}$-valued. Similar arguments apply to the other modalities.

Given the formal framework as defined above, we modify the definition of Floyd's inductive assertion network as follows in order to obtain a method of proving absence of runtime errors in φ of transition diagrams P.

Definition 6.5 (Floyd's inductive assertion method including absence of runtime errors)

1. Consider a total $\{tt, ff\}$-valued boolean function φ.
2. Choose the predicates $\mathcal{Q}_l$ to be total and $\{tt, ff\}$-valued.
3. Prove that $\mathcal{Q}$ is inductive *and implies freedom from runtime errors* of P, that is, for each transition (l, a, l') of P, prove the validity of

$$\models \mathcal{Q}_l \wedge c \rightarrow (\mathcal{Q}_{l'} \circ f) \wedge Def(f),$$

with $Def(f)$ as above, assuming that $a = c \rightarrow f$.

4. Prove that $\mathcal{Q}$ is consistent w.r.t. precondition φ, i.e., require the proof of $\models \varphi \rightarrow \mathcal{Q}_s$. □

This method verifies explicitly whether the applied state transformations in P are well-defined. Consequently, soundness of the method is obvious. Its semantic completeness can be proved using the reachability predicates $\mathcal{Q}_l$ from Theorem 4.4 as the assertion network.

References

[dRE98] W.-P. de Roever and K. Engelhardt. *Data Refinement*. Cambridge University Press, 1998.