程序验证方法

研究生课程

Chapter 3 (3.1, 3.2, 3.4)

while Programs

朱惠彪

华东师范大学 软件学院

Contents

Syntax

$$S ::= skip \mid u{:=}t \mid S_1 ; S_2$$
$$\mid if\ B\ then\ S_1\ else\ S_2\ fi \mid while\ B\ do\ S_1\ od$$

$var(S)$: *the set of all simple and array variables that appear in* S.

$change(S)$: *the set of all simple and array variables that can be modified by* S.

Semantics

- *A mapping* $\mathcal{M}[\![S]\!]$ *from proper(initial) states to final states*
- ***Configuration:*** $< S, \sigma >$
- ***Transition:*** $< S, \sigma > \rightarrow < R, \tau >$
- ***Empty program:*** ***E***

$\boxed{R \equiv E \text{ means that } S \text{ terminates in } \tau}$

Semantics

- $\boldsymbol{Transition\ axioms\ and\ rules(\sigma\ is\ a\ proper\ state)}$

(i) $< skip, \sigma > \ \rightarrow \ < E, \sigma >$,

(ii) $< u := t, \sigma > \ \rightarrow \ < E, \sigma[u := \sigma(t)] >$,

(iii) $$\frac{< S_1, \sigma > \ \rightarrow \ < S_2, \tau >}{< S_1;\ S, \sigma > \ \rightarrow \ < S_2;\ S, \tau >},$$

(iv) $< \textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}, \sigma > \ \rightarrow \ < S_1, \sigma >$ where $\sigma \models B$,

(v) $< \textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}, \sigma > \ \rightarrow \ < S_2, \sigma >$ where $\sigma \models \neg B$,

(vi) $< \textbf{while } B \textbf{ do } S \textbf{ od}, \sigma > \ \rightarrow \ < S;\ \textbf{while } B \textbf{ do } S \textbf{ od}, \sigma >$
where $\sigma \models B$,

(vii) $< \textbf{while } B \textbf{ do } S \textbf{ od}, \sigma > \ \rightarrow \ < E, \sigma >$, where $\sigma \models \neg B$.

Semantics

Definition 3.1 $(S\ is\ a\ while\ program\ and\ \sigma\ is\ a\ proper\ state)$

(1) ***transition sequence :*** ***a finite or infinite sequence of configuration*** $< S_i, \sigma_i > (i \geq 0)$ ***such that***

$$< S, \sigma > = < S_0, \sigma_0 > \rightarrow < S_1, \sigma_1 > \rightarrow \ldots \rightarrow < S_i, \sigma_i > \rightarrow \ldots$$

Semantics

(2) computation : a transition sequence of S starting in σ .

A computation terminates in τ if it is finite and its last configuration is of the form $< E, \tau >$.

A computation diverges if it is infinite.

Semantics

- *(3)* To describe the effect of finite transition sequences we use the transitive, reflexive closure $\rightarrow^*$ of the transition relation $\rightarrow$:

$$< S, \sigma > \;\rightarrow^*\; < R, \tau >$$

holds when there exist configurations $< S_1, \sigma_1 >, \ldots, < S_n, \sigma_n >$ with $n \geq 0$ such that

$$< S, \sigma > = < S_1, \sigma_1 > \;\rightarrow \ldots \rightarrow\; < S_n, \sigma_n > = < R, \tau >$$

holds. In the case when $n = 0$, $< S, \sigma > = < R, \tau >$ holds.

Semantics

- ***Deterministic programs***

Lemma 3.1. (Determinism) *For any* **while** *program S and a proper state σ, there is exactly one computation of S starting in σ.*

- ***If S did not terminate then it can be executed for at least one step***

Lemma 3.2. (Absence of Blocking) *If $S \not\equiv E$ then for any proper state σ there exists a configuration $< S_1, \tau >$ such that*

$$< S, \sigma > \; \rightarrow \; < S_1, \tau > .$$

Semantics

Definition 3.2. We now define two input/output semantics for **while** programs. Each of them associates with a program S and a proper state $\sigma \in \Sigma$ a set of output states.

(i) The *partial correctness semantics* is a mapping

$$\mathcal{M}[\![S]\!] : \Sigma \to \mathcal{P}(\Sigma)$$

with

$$\mathcal{M}[\![S]\!](\sigma) = \{\tau \mid < S, \sigma > \to^* < E, \tau >\}.$$

(ii) The *total correctness semantics* is a mapping

$$\mathcal{M}_{tot}[\![S]\!] : \Sigma \to \mathcal{P}(\Sigma \boxed{\cup \{\bot\}})$$

$\bot$ *indicates divergence*

with

$$\mathcal{M}_{tot}[\![S]\!](\sigma) = \mathcal{M}[\![S]\!](\sigma) \boxed{\cup \{\bot \mid S \text{ can diverge from } \sigma\}.}$$

Semantics

Example 3.1. Consider the program

$$S \equiv a[0] := 1;\ a[1] := 0;\ \textbf{while}\ a[x] \neq 0\ \textbf{do}\ x := x + 1\ \textbf{od}$$

and let σ be a proper state in which x is 0.

$$\begin{aligned}
& < S, \sigma > \\
\rightarrow\ & < a[1] := 0;\ \textbf{while}\ a[x] \neq 0\ \textbf{do}\ x := x + 1\ \textbf{od}, \sigma[a[0] := 1] > \\
\rightarrow\ & < \textbf{while}\ a[x] \neq 0\ \textbf{do}\ x := x + 1\ \textbf{od}, \boxed{\sigma'} > \qquad \boxed{\sigma' = \sigma[a[0] := 1][a[1] := 0]} \\
\rightarrow\ & < x := x + 1;\ \textbf{while}\ a[x] \neq 0\ \textbf{do}\ x := x + 1\ \textbf{od}, \sigma' > \\
\rightarrow\ & < \textbf{while}\ a[x] \neq 0\ \textbf{do}\ x := x + 1\ \textbf{od}, \sigma'[x := 1] > \\
\rightarrow\ & < E, \sigma'[x := 1] > .
\end{aligned}$$

Thus S when activated in σ terminates in five steps. We have

$$\boxed{\mathcal{M}[\![S]\!](\sigma) = \mathcal{M}_{tot}[\![S]\!](\sigma) = \{\sigma'[x := 1]\}.}$$

Now let τ be a state in which x is 2 and for $i = 2, 3, \ldots, a[i]$ is 1. The computation of S starting in τ has the following form where τ' stands for $\tau[a[0] := 1][a[1] := 0]$:

$$\boxed{S \equiv a[0] := 1;\ a[1] := 0;\ \textbf{while } a[x] \neq 0 \textbf{ do } x := x + 1 \textbf{ od}}$$

$$
\begin{aligned}
& < S, \tau > \\
\rightarrow\ & < a[1] := 0;\ \textbf{while } a[x] \neq 0 \textbf{ do } x := x + 1 \textbf{ od}, \tau[a[0] := 1] > \\
\rightarrow\ & < \textbf{while } a[x] \neq 0 \textbf{ do } x := x + 1 \textbf{ od}, \tau' > \\
\rightarrow\ & < x := x + 1;\ \textbf{while } a[x] \neq 0 \textbf{ do } x := x + 1 \textbf{ od}, \tau' > \\
\rightarrow\ & < \textbf{while } a[x] \neq 0 \textbf{ do } x := x + 1 \textbf{ od}, \tau'[x := \tau(x) + 1] > \\
& \ldots \\
\rightarrow\ & < \textbf{while } a[x] \neq 0 \textbf{ do } x := x + 1 \textbf{ od}, \tau'[x := \tau(x) + k] > \\
& \ldots
\end{aligned}
$$

Thus S can diverge from τ. We have $\boxed{\mathcal{M}[\![S]\!](\tau) = \emptyset \text{ and } \mathcal{M}_{tot}[\![S]\!](\tau) = \{\perp\}.}$

$\square$

Semantics

Properties of Semantics

- *Notations:*
  - ✓ Ω: $a\ while\ program\ such\ that\ for\ all\ proper\ states\ \sigma$

$$\mathcal{M}[\![\Omega]\!](\sigma) = \emptyset; \qquad \mathcal{M}_{tot}[\![\Omega]\!](\sigma) = \{\perp\}$$

  - ✓ N: $stands\ for$ M or M_{tot} $\boxed{\mathcal{M}[\![S]\!](\perp) = \emptyset \text{ and } \mathcal{M}_{tot}[\![S]\!](\perp) = \{\perp\}}$

$$\mathcal{N}[\![S]\!](X) = \bigcup_{\sigma \in X} \mathcal{N}[\![S]\!](\sigma).$$

$$X \subseteq \Sigma \cup \{\perp\}$$

Semantics

$$\begin{aligned}(\text{while } B \text{ do } S \text{ od})^0 &= \Omega,\\ (\text{while } B \text{ do } S \text{ od})^{k+1} &= \text{if } B \text{ then } S;\ (\text{while } B \text{ do } S \text{ od})^k\\ &\quad\ \text{else } skip \text{ fi}.\end{aligned}$$

Lemma 3.3. (Input/Output)

(i) $\mathcal{N}[\![S]\!]$ *is monotonic; that is,* $X \subseteq Y \subseteq \Sigma \cup \{\perp\}$ *implies* $\mathcal{N}[\![S]\!](X) \subseteq \mathcal{N}[\![S]\!](Y)$.

(ii) $\mathcal{N}[\![S_1;\ S_2]\!](X) = \mathcal{N}[\![S_2]\!](\mathcal{N}[\![S_1]\!](X))$.

(iii) $\mathcal{N}[\![(S_1;\ S_2);\ S_3]\!](X) = \mathcal{N}[\![S_1;\ (S_2;\ S_3)]\!](X)$.

(iv) $\mathcal{N}[\![\textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}]\!](X) =$
$\mathcal{N}[\![S_1]\!](X \cap [\![B]\!]) \cup \mathcal{N}[\![S_2]\!](X \cap [\![\neg B]\!]) \cup \{\perp \mid \perp \in X \textit{ and } \mathcal{N} = \mathcal{M}_{tot}\}$.

$$[\![p]\!] = \{\sigma \mid \sigma \text{ is a proper state and } \sigma \models p\}.$$

(v) $\mathcal{M}[\![\textbf{while } B \textbf{ do } S \textbf{ od}]\!] = \bigcup_{k=0}^{\infty} \mathcal{M}[\![(\textbf{while } B \textbf{ do } S \textbf{ od})^k]\!]$.

Semantics

Lemma 3.4. (Change and Access)

(i) For all proper states σ *and* τ, $\tau \in \mathcal{N}[\![S]\!](\sigma)$ *implies*

$$\tau[Var - change(S)] = \sigma[Var - change(S)].$$

(ii) For all proper states σ *and* τ, $\sigma[var(S)] = \tau[var(S)]$ *implies*

$$\mathcal{N}[\![S]\!](\sigma) = \mathcal{N}[\![S]\!](\tau) \textbf{ mod } Var - var(S).$$

(i) ***every program S changes at most the variables in change(S)***

(ii) ***every program S accesses at most the variables in var(S)***

Note: We say that two sets of states X and Y *agree modulo* Z, and write **X = Y mod Z**, if **{σ[Var − Z] | σ ∈ X} = {σ[Var − Z] | σ ∈ Y }**. (**page 34**)

Verification

- **Please see a separate file.**

3.4 Proof Outlines

- Formal proofs are long and tedious to follow.
- It is better to organize the proof in small local isolated steps
- We can use the structure of the program to structure our proof!

The idea

- For the program $P = c_1; c_2; c_3; \dots c_n$ we want to show

$$\vdash \{\phi_0\}P\{\phi_n\}$$

- We can split the problem into smaller ones if we find formulas ϕ_i's such that

$$\vdash \{\phi_i\}c_i\{\phi_{i+1}\}$$

The idea

- **Thus we have to find a calculus for presenting a proof $\vdash \{\boldsymbol{\phi_0}\}\boldsymbol{P}\{\boldsymbol{\phi_n}\}$ by interleaving formulas with code**

$$
\begin{array}{c}
\{\phi_0\} \\
\hline
c_1; \\
\{\phi_1\} \\
c_2; \\
\{\phi_2\} \\
c_3; \\
\vdots \\
\{\phi_{n-1}\} \\
c_n \\
\{\phi_n\} \\
\hline
\end{array}
$$

Proof Outlines

◆*Partial Correctness*

➢ $\boldsymbol{Definition\ 3.6.\ (Proof\ Outline: Partial\ Correctness)}$

- $\boldsymbol{Let\ S^{*}\ stand\ for\ the\ program\ S\ annotated,}$

  $\boldsymbol{with\ assertions\ ,some\ of\ them\ \ labeled\ by\ the\ keyword\ inv.}$

- $\boldsymbol{We\ define\ the\ notion\ of\ a\ proof\ outline\ for\ partial\ correctness}$

  $\boldsymbol{inductively\ by\ the\ formation\ axioms\ and\ rules.}$

Proof Outlines

- $\boldsymbol{A\ formation\ axiom\ \varphi\ is\ a\ proof\ outline}$.
- $\boldsymbol{A\ formation\ rule}$

$$\frac{\varphi_1, \ldots, \varphi_k}{\varphi_{k+1}}$$

$\boldsymbol{If\ \varphi_1, \ldots, \varphi_k\ are\ proof\ outlines, then\ \varphi_{k+1}\ is\ a\ proof\ outline}$.

Proof Outlines

(i) $\{p\}\ skip\ \{p\}$

(ii) $\{p[u := t]\}\ u := t\ \{p\}$

(iii) $$\frac{\{p\}\ S_1^*\ \{r\}, \{r\}\ S_2^*\ \{q\}}{\{p\}\ S_1^*;\ \{r\}\ S_2^*\ \{q\}}$$

(iv) $$\frac{\{p \wedge B\}\ S_1^*\ \{q\}, \{p \wedge \neg B\}\ S_2^*\ \{q\}}{\{p\}\ \textbf{if}\ B\ \textbf{then}\ \{p \wedge B\}\ S_1^*\ \{q\}\ \textbf{else}\ \{p \wedge \neg B\}\ S_2^*\ \{q\}\ \textbf{fi}\ \{q\}}$$

(v) $$\frac{\{p \wedge B\}\ S^*\ \{p\}}{\boxed{\{\textbf{inv} : p\}}\ \textbf{while}\ B\ \textbf{do}\ \{p \wedge B\}\ S^*\ \{p\}\ \textbf{od}\ \{p \wedge \neg B\}}$$

(vi) $$\frac{p \rightarrow p_1,\ \{p_1\}\ S^*\ \{q_1\},\ q_1 \rightarrow q}{\{p\}\{p_1\}\ S^*\ \{q_1\}\{q\}}$$

(vii) $$\frac{\{p\}\ S^*\ \{q\}}{\{p\}\ S^{**}\ \{q\}}$$

where S^{**} results from S^* by omitting some annotations of the form $\{r\}$. Thus all annotations of the form $\{\textbf{inv} : r\}$ remain.

Proof Outlines

- $\boldsymbol{If\ every\ subprogram\ T\ of\ S\ is\ preceded\ by}$
$\boldsymbol{exactly\ one\ assertion\ \ in\ S^*\ called\ pre(T),}$
$\boldsymbol{then\ a\ proof\ outline\ \{p\}\ S^*\{q\}\ for\ partial\ correctness}$
$\boldsymbol{is\ called\ \underline{standard}.}$

Proof Outlines

Theorem 3.2.

(i) Let $\{p\}\ S^*\ \{q\}$ *be a proof outline for partial correctness. Then* $\vdash_{PD} \{p\}\ S\ \{q\}$.

(ii) If $\vdash_{PD} \{p\}\ S\ \{q\}$, *there exists a standard proof outline for partial correctness of the form* $\{p\}\ S^*\ \{q\}$.

$\vdash_{PD}$ ***stands for provability in the system PW augmented by the set of all true assertions.***

Proof Outlines

Example 3.6. Let us reconsider the integer division program studied in Example 3.4. We present the correctness formulas (3.5), (3.6) and (3.7) in the following form:

$\{x \geq 0 \wedge y \geq 0\}$
$quo := 0;\ rem := x;$
$\boxed{\{\mathbf{inv} : p\}}$
$\mathbf{while}\ rem \geq y\ \mathbf{do}$
$\quad \{p \wedge rem \geq y\}$
$\quad rem := rem - y;\ quo := quo + 1$
$\mathbf{od}$
$\{p \wedge rem < y\}$
$\{quo \cdot y + rem = x \wedge 0 \leq rem < y\},$

label the loop invariant
$\boldsymbol{p \equiv quo \cdot y + rem = x \wedge rem \geq 0}$

$\{q_1\}\{q_2\}$ *stand for the implication* $q_1 \rightarrow q_2$ *is true*

Proof Outlines

- $The\ proof\ outlines\ \{p\}\ S^*\ \{q\}$

$enjoy\ the\ following\ useful\ and\ intuitive\ property$:

whenever the control of S in a given computation *starting in*

***a state satisfying p* reaches a point annoated by an assertion,**

this assertion is true.

Proof Outlines

- $\boldsymbol{at(T, S)}$: ***the remainder of S that is to be executed***

  when the control is at subprogram T.

- ***Example***

$S \equiv \textbf{while } x \geq 0 \textbf{ do if } y \geq 0 \textbf{ then } x := x - 1 \textbf{ else } \boxed{y := y - 2} \textbf{ fi od},$

$T \equiv y := y - 2,$

$$\boxed{\textbf{at}(T, S) \equiv \textbf{at}(y := y-2, S) \equiv y := y-2; S}$$

Proof Outlines

➢ $\boldsymbol{Definition\ 3.7.\ T\ is\ a\ subprogram\ of\ S. We\ define\ a\ program}$

$\boldsymbol{at(T,S)\ by\ the\ following\ clauses}$:

(i) if $S \equiv S_1;\ S_2$ and T is a subprogram of S_1, then $\mathbf{at}(T, S) \equiv \mathbf{at}(T;\ S_1);$ S_2 and if T is a subprogram of S_2 then $\mathbf{at}(T, S) \equiv \mathbf{at}(T, S_2)$;

(ii) if $S \equiv$ **if** B **then** S_1 **else** S_2 **fi** and T is a subprogram of S_i, then $\mathbf{at}(T, S) \equiv \mathbf{at}(T, S_i)$ $(i = 1, 2)$;

(iii) if $S \equiv$ **while** B **do** S' **od** and T is a subprogram of S', then $\mathbf{at}(T, S)$ $\equiv \underline{\mathbf{at}(T, S');\ S}$;

(iv) if $T \equiv S$ then $\mathbf{at}(T, S) \equiv S$. □

Proof Outlines

Theorem 3.3. (Strong Soundness) *Let* $\{p\}\ S^*\ \{q\}$ *be a standard proof outline for partial correctness. Suppose that*

$$< S, \sigma > \ \rightarrow^* \ < R, \tau >$$

for some state σ *satisfying* p*, program* R *and state* τ*. Then*

- *if* $R \equiv \mathbf{at}(T, S)$ *for a subprogram* T *of* S*, then* $\tau \models pre(T)$*,*
- *if* $R \equiv E$ *then* $\tau \models q$*.*

Proof Outlines

◆ *Total Correctness*

➢ $\boldsymbol{Definition\ 3.8.\ (Proof\ Outline: Total\ Correctness)}$

- $\boldsymbol{Let\ S^{*}\ and\ S^{**}\ stand\ for\ the\ program\ S\ annotated,}$

  $\boldsymbol{with\ assertions\ ,some\ of\ them\ labeled\ by\ the\ keyword\ inv,}$

  $\boldsymbol{and\ integer\ expressions, all\ labled\ by\ the\ keyword\ bd\ .}$

- $\boldsymbol{The\ notion\ of\ a\ proof\ outline\ for\ total\ correctness\ is\ defined\ as\ for\ partial\ correctness,}$

  $\boldsymbol{except\ for\ formation\ rule\ (v)\ dealing\ with\ loops.}$

Proof Outlines

$$\frac{\{p \wedge B\}\ S\ \{p\},\quad \{p \wedge B \wedge t = z\}\ S\ \{t < z\},\quad p \to t \geq 0}{\{p\}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}\ \{p \wedge \neg B\}}$$

$$\text{(v)}\quad \frac{\{p \wedge B\}\ S^*\ \{p\}}{\{\textbf{inv} : p\}\ \textbf{while}\ B\ \textbf{do}\ \{p \wedge B\}\ S^*\ \{p\}\ \textbf{od}\ \{p \wedge \neg B\}}$$

replaced by

(viii)

$$\frac{\{p \wedge B\}\ S^*\ \{p\},\quad \boxed{\{p \wedge B \wedge t = z\}\ S^{**}\ \{t < z\},\quad p \to t \geq 0}}{\{\textbf{inv} : p\}\boxed{\{\textbf{bd} : t\}}\ \textbf{while}\ B\ \textbf{do}\ \{p \wedge B\}\ S^*\ \{p\}\ \textbf{od}\ \{p \wedge \neg B\}}$$

$\{\textbf{bd}: t\}$ *represents the bound function of the loop* ***while B do S od***.

where t is an integer expression and z is an integer variable not occurring in p, t, B or S^{**}.

Proof Outlines

Example 3.7. The following is a proof outline for total correctness of the integer division program DIV studied in Example 3.4:

$$
\begin{array}{l}
\{x \geq 0 \land y > 0\} \\
quo := 0;\ rem := x; \\
\boxed{\{\mathbf{inv} : p'\}\{\mathbf{bd} : rem\}} \\
\mathbf{while}\ rem \geq y\ \mathbf{do} \\
\qquad \{p' \land rem \geq y\} \\
\qquad rem := rem - y;\ quo := quo + 1 \\
\qquad \{p'\} \\
\mathbf{od} \\
\{p' \land rem < y\} \\
\{quo \cdot y + rem = x \land 0 \leq rem < y\},
\end{array}
$$

where

$$p' \equiv quo \cdot y + rem = x \land rem \geq 0 \land y > 0.$$

Thank You!