Formal Semantics

Yixiang Chen

East China Normal University

Outline

- **IMP** Language
- Operational Semantics of **IMP**
- Denotational Semantics of **IMP**
- Axiomatic Semantics of **IMP**
- UTP

IMP Language

IMP Language

- Numbers **N**, consisting of positive and negative integers with zero, n, m range over numbers **N**,
- truth values **T**=$\{\text{true}, \text{fals}\}$
- locations **Loc**, X, Y range over locations,
- arithmetic expression **Aexp**, a ranges over arithmetics expressions,
- boolean expressions **Bexp**, b ranges over boolean expressions
- commands **Com**, c ranges over commands

IMP Laguange

The formation rules of the whole of IMP

- **Aexp**:

$$a ::= n \mid X \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$$

- **Bexp**:

$$b ::= \text{true} \mid \text{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \neg b \mid b_0 \wedge b_1 \mid b_0 \vee b_1$$

- **Com**:

$$c ::= \text{skip} \mid X := a \mid c_0; c_1 \mid \text{if } b \text{ then } c_0 \text{ else } c_1 \mid \text{while } b \text{ do } c$$

Evaluation of **IMP**

- States: a functions $\sigma : \mathbf{Loc} \rightarrow \mathbf{N}$ from locations to numbers.
  $\sigma(X)$ is the value, or contents, of locations X in state σ
- Σ: The set of states.

Evaluation of **IMP**

The evaluation of arithmetic expressions a in a state σ:

- **Numbers** n:

$$\frac{}{< n, \sigma > \longrightarrow n}$$

- **Locations**:

$$\frac{}{< X, \sigma > \longrightarrow \sigma(X)}$$

- **Sum, Subtraction, Products**:

$$\frac{< a_0, \sigma > \longrightarrow n_0 \quad < a_1, \sigma > \longrightarrow n_1}{< a_0 * a_1, \sigma > \longrightarrow n_0 * n_1}$$

Evaluation of IMP

The evaluation of boolean expressions: Evaluate boolean expressions to truth values (true, false):

$$\frac{}{< \mathsf{true}, \sigma > \to \mathsf{true}} \qquad \frac{}{< \mathsf{false}, \sigma > \to \mathsf{false}}$$

$$\frac{< a_0, \sigma > \to n \quad < a_1, \sigma > \to m}{< a_0 = a_1, \sigma > \to \mathsf{true}} \quad \text{if } n = m$$

$$\frac{< a_0, \sigma > \to n \quad < a_1, \sigma > \to m}{< a_0 = a_1, \sigma > \to \mathsf{false}} \quad \text{if } n \neq m$$

$$\frac{< a_0, \sigma > \to n \quad < a_1, \sigma > \to m}{< a_0 \leq a_1, \sigma > \to \mathsf{true}} \quad \text{if } n \leq m$$

$$\frac{< a_0, \sigma > \to n \quad < a_1, \sigma > \to m}{< a_0 \leq a_1, \sigma > \to \mathsf{false}} \quad \text{if } n \not\leq m$$

Evaluation of IMP

The evaluation of boolean expressions: Evaluate boolean expressions to truth values (**true**, **false**):

$$\frac{< b, \sigma > \rightarrow \text{true}}{< \neg b, \sigma > \rightarrow \text{false}} \qquad \frac{< b, \sigma > \rightarrow \text{false}}{< \neg b, \sigma > \rightarrow \text{true}}$$

$$\frac{< b_0, \sigma > \rightarrow t_0 \quad < b_1, \sigma > \rightarrow t_1}{< b_0 \wedge b_1, \sigma > \rightarrow t_0 \wedge t_1} \qquad \frac{< b_0, \sigma > \rightarrow t_0 \quad < b_1, \sigma > \rightarrow t_1}{< b_0 \vee b_1, \sigma > \rightarrow t_0 \vee t_1}$$

Operational Semantics of **IMP**

Operational Semantics of IMP

Atomic commands

$$\frac{}{< \mathsf{skip}, \sigma > \longrightarrow \sigma}$$

$$\frac{< a, \sigma > \longrightarrow m}{< X := a, \sigma > \longrightarrow \sigma[m/X]}$$

Sequencing

$$\frac{< c_0, \sigma > \longrightarrow \sigma'' \quad < c_1, \sigma'' > \longrightarrow \sigma'}{< c_0; c_1, \sigma > \longrightarrow \sigma'}$$

Conditionals

$$\frac{< b, \sigma > \longrightarrow \mathsf{true} \quad < c_0, \sigma > \longrightarrow \sigma'}{< \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1, \sigma > \longrightarrow \sigma'}$$

$$\frac{< b, \sigma > \longrightarrow \mathsf{false} \quad < c_1, \sigma > \longrightarrow \sigma'}{< \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1, \sigma > \longrightarrow \sigma'}$$

While – loops

$$\frac{< b, \sigma > \longrightarrow \mathsf{false}}{< \textbf{while } b \textbf{ do } c, \sigma > \longrightarrow \sigma}$$

$$\frac{< b, \sigma > \longrightarrow \mathsf{true}, < c, \sigma > \longrightarrow \sigma'' \quad < \textbf{while } b \textbf{ do } c, \sigma'' > \longrightarrow \sigma'}{< \textbf{while } b \textbf{ do } c, \sigma > \longrightarrow \sigma'}$$

IMP计算

基于上面的**IMP**操作语义，我们可以建立**IMP**语言程序的计算概念。

Definition

IMP程序计算关系⇒是指一从集合$\mathbf{Com} \times \Sigma$到集合$(\mathbf{Com} \times \Sigma) \cup \Sigma$的关系. 设$c, c' \in \mathbf{Com}, \sigma, \sigma' \in \Sigma$, 定义$< c, \sigma > \Rightarrow < c', \sigma' > | \sigma'$，若下面三条中之一成立

1. $c = c', \sigma = \sigma'$
2. 若c是单个命令，且$< c, \sigma > \rightarrow \sigma'$, 则$< c, \sigma > \Rightarrow \sigma'$
3. 若c是单个命令c_1和程序c_2的顺序序列，即$c = c_1; c_2$,并且$< c_1, \sigma > \rightarrow \sigma''$，$< c_2, \sigma'' > \Rightarrow < c', \sigma' >$。

IMP计算

Definition

设$c \in \mathbf{Com}, \sigma \in \Sigma$，若存在$\sigma' \in \Sigma$使得$< c, \sigma > \Rightarrow \sigma'$, 则称程序$c$在状态$\sigma$下终止，$\sigma'$为$c$在$\sigma$处计算的结果。

IMP计算：例子

(1) 设程序c_1为$X := (X+1) \times (Y+Z); Y := (X \times X + Y \times Y + 2 \times X \times Y); Z := X + Y - X \times Y$. 再设状态$\sigma$为: $\sigma(X) = 8, \sigma(Y) = -2, \sigma(Z) = 3$, 计算$c_1$在状态$\sigma$处计算的结果。

(2) 设程序c_2为$X := X \times Y;\ \textbf{if}\ X \leq Y\ \textbf{then}\ Y := (Y - X) \times (Z \times Y - X)\ \textbf{else}\ Z := Z \times Y + X$, 计算程序$c_2$在下面状态$\sigma$处计算的结果:

1. $\sigma_1(X) = 6, \sigma_1(Y) = 100, \sigma_1(Z) = -10$
2. $\sigma_2(X) = 100, \sigma_2(Y) = 10, \sigma_2(Z) = -10$
3. $\sigma_3(X) = 100, \sigma_3(Y) = 100, \sigma_3(Z) = 100$

IMP计算：例子

(3) 设程序c_3为$X := X + Y - Z;$ **if** $X + 1 \leq X \times Y$ **then** $X := (X + 1) \times Z; Y := (X + Y) \times (Z - 1)$ **else** $(Z := X \times Y - 1; X := Y \times Z);$ **while** $X \leq 1000$ **do** $(Y := X + Y, Z := X \times Z; X := X + 2); X := Y; Y := Z; Z := X$, 设状态$\sigma$为: $\sigma(X) = 2, \sigma(Y) = 10, \sigma(Z) = 6$,计算程序$c_3$在此状态处计算的结果。

Natural Equivalence Relation on Commands

Definition

$C_0 \sim c_1$ *iff* $\forall \sigma, \sigma' \in \Sigma. < c_0, \sigma > \rightarrow \sigma'$ *iff* $< c_1, \sigma > \rightarrow \sigma'$.

Proposition

Let $w \equiv \textbf{while } b \textbf{ do } c$. *Then*

$$w \sim \textbf{if } b \textbf{ then } c; w \textbf{ else } \mathsf{skip}.$$

Natural Induction on Derivation

Definition

$\models < c, \sigma > \rightarrow \sigma$ *Meaning* $< c, \sigma > \rightarrow \sigma'$ *is derivable from the operational semantics of commands*

Proposition

Let $w \equiv \textbf{while } b \textbf{ do } c$. *Then*

$$w \sim \textbf{if } b \textbf{ then } c; w \textbf{ else } \mathsf{skip}.$$

Deterministic of Execution of Commands

Theorem

Let c be a command and σ_0 a state. If $< c, \sigma_0 > \longrightarrow \sigma_1$ and $< c, \sigma_0 > \longrightarrow \sigma$ then $\sigma = \sigma_1$,for all states σ, σ_1.

$$c ::= \text{skip} \mid X := a \mid c_0; c_1 \mid \text{if } b \text{ then } c_0 \text{ else } c_1 \mid \text{while } b \text{ do } c$$

Denotational Semantics of **IMP**

Denotational Semantics of **IMP**

Definition

we define the semantic function as a relation by structural induction:

$$
\begin{aligned}
\mathcal{A} &: \mathbf{Aexp} \to \mathcal{P}(\Sigma \times \mathbf{N}) \\
\mathcal{A} &: \mathbf{Aexp} \to (\Sigma \to \mathbf{N}) \\
\mathcal{B} &: \mathbf{Bexp} \to \mathcal{P}(\Sigma \times \mathbf{T}) \\
\mathcal{B} &: \mathbf{Bexp} \to (\Sigma \to \mathbf{T}) \\
\mathcal{C} &: \mathbf{Com} \to \mathcal{P}(\Sigma \times \Sigma) \\
\mathcal{C} &: \mathbf{Com} \to (\Sigma \to \Sigma)
\end{aligned}
$$

Denotational Semantics of **Aexp**

Definition

we define the semantic function by structural induction:

$$\mathcal{A}[\![n]\!] = \{(\sigma, n) \mid \sigma \in \Sigma\}$$

$$\mathcal{A}[\![X]\!] = \{(\sigma, \sigma(X)) \mid \sigma \in \Sigma\}$$

$$\mathcal{A}[\![a_0 + a_1]\!] = \{\sigma, n_0 + n_1) \mid (\sigma, n_0) \in \mathcal{A}[\![a_0]\!] \& (\sigma, n_1) \in \mathcal{A}[\![a_1]\!]\}$$

$$\mathcal{A}[\![a_0 - a_1]\!] = \{(\sigma, n_0 - n_1) \mid (\sigma, n_0) \in \mathcal{A}[\![n_0]\!] \& (\sigma, n_1) \in \mathcal{A}[\![n_1]\!]\}$$

$$\mathcal{A}[\![a_0 \times a_1]\!] = \{(\sigma, n_0 \times n_1) \mid (\sigma, n_0) \in \mathcal{A}[\![n_0]\!] \& (\sigma, n_1) \in \mathcal{A}[\![n_1]\!]\}$$

$$a ::= n \mid X \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$$

Denotational Semantics of **Bexp**

Definition

$$
\begin{aligned}
\mathcal{B}[\![\mathbf{true}]\!] &= \{(\sigma, \mathbf{true}) \mid \sigma \in \Sigma\} \\
\mathcal{B}[\![\mathbf{false}]\!] &= \{(\sigma, \mathbf{false}) \mid \sigma \in \Sigma\} \\
\mathcal{B}[\![a_0 = a_1]\!] &= \{(\sigma, \mathbf{true}) \mid \sigma \in \Sigma \& \mathcal{A}[\![a_0]\!]\sigma = \mathcal{A}[\![a_1]\!]\sigma\} \\
&\quad \cup \{(\sigma, \mathbf{false}) \mid \sigma \in \Sigma \& \mathcal{B}[\![a_0]\!]\sigma \neq \mathcal{A}[\![a_1]\!]\sigma\} \\
\mathcal{B}[\![a_0 \leq a_1]\!] &= \{(\sigma, \mathbf{true}) \mid \sigma \in \Sigma \& \mathcal{A}[\![a_0]\!]\sigma \leq \mathcal{A}[\![a_1]\!]\sigma\} \\
&\quad \cup \{(\sigma, \mathbf{false}) \mid \sigma \in \Sigma \& \mathcal{B}[\![a_0]\!]\sigma \not\leq \mathcal{A}[\![a_1]\!]\sigma\}
\end{aligned}
$$

$$b ::= \text{true} \mid \text{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \neg b \mid b_0 \wedge b_1 \mid b_0 \vee b_1$$

Denotational Semantics of **Bexp**-contiu.

Definition

we define the semantic function by structural induction:

$$\mathcal{B}[\![\neg b]\!] = \{(\sigma, \neg_T t) \mid \sigma \in \mathcal{B}[\![b]\!]\}$$

$$\mathcal{B}[\![b_0 \wedge b_1]\!] = \{(\sigma, t_0 \wedge_T t_1) \mid \sigma \in \Sigma \& (\sigma, t_0) \in \mathcal{B}[\![b_0]\!] \& (\sigma, t_1) \in \mathcal{B}[\![b_1]\!]\}$$

$$\mathcal{B}[\![b_0 \vee b_1]\!] = \{(\sigma, t_0 \vee_T t_1) \mid \sigma \in \Sigma \& (\sigma, t_0) \in \mathcal{B}[\![b_0]\!] \& (\sigma, t_1) \in \mathcal{B}[\![b_1]\!]\}$$

$$b ::= \text{true} \mid \text{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \neg b \mid b_0 \wedge b_1 \mid b_0 \vee b_1$$

Denotational Semantics of **Bexp**-summary

- $\mathcal{B}[\![\mathbf{true}]\!]\sigma = \mathbf{true}$
- $\mathcal{B}[\![\mathbf{false}]\!]\sigma = \mathbf{false}$
- $\mathcal{B}[\![a_0 = a_1]\!]\sigma = \begin{cases} \mathbf{true} & \text{if } \mathcal{A}[\![a_0]\!]\sigma = \mathcal{A}[\![a_1]\!]\sigma \\ \mathbf{false} & \text{if } \mathcal{A}[\![a_0]\!]\sigma \neq \mathcal{A}[\![a_1]\!]\sigma \end{cases}$
- $\mathcal{B}[\![a_0 \leq a_1]\!]\sigma = \begin{cases} \mathbf{true} & \text{if } \mathcal{A}[\![a_0]\!]\sigma \leq \mathcal{A}[\![a_1]\!]\sigma \\ \mathbf{false} & \text{if } \mathcal{A}[\![a_0]\!]\sigma \not\leq \mathcal{A}[\![a_1]\!]\sigma \end{cases}$
- $\mathcal{B}[\![\neg b]\!]\sigma = \neg_T \mathcal{B}[\![b]\!]\sigma$
- $\mathcal{B}[\![b_0 \wedge b_1]\!]\sigma = \mathcal{B}[\![b_0]\!]\sigma \wedge_T \mathcal{B}[\![b_1]\!]\sigma$
- $\mathcal{B}[\![b_0 \vee b_1]\!]\sigma = \mathcal{B}[\![b_0]\!]\sigma \vee_T \mathcal{B}[\![b_1]\!]\sigma$

Denotational Semantics of **Com**

Definition

$$\mathcal{C}[\![\mathbf{skip}]\!] = \{(\sigma,\sigma) \mid \sigma \in \blacksquare\}$$

$$\mathcal{C}[\![X := a]\!] = \{(\sigma, \sigma[n/X]) \mid \sigma \in \Sigma \;\&\; n = \mathcal{A}[\![a]\!]\sigma\}$$

$$\mathcal{C}[\![c_0; c_1]\!] = \mathcal{C}[\![c_1]\!] \circ \mathcal{C}[\![c_0]\!]$$

$$\mathcal{C}[\![\mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1]\!] =$$

$$\{(\sigma,\sigma') \mid \mathcal{B}[\![\mathbf{b}]\!]\sigma = \mathbf{true} \;\&\; (\sigma,\sigma') \in \mathcal{C}[\![\mathbf{c_0}]\!]\} \cup$$

$$\{(\sigma,\sigma') \mid \mathcal{B}[\![\mathbf{b}]\!]\sigma = \mathbf{false} \& (\sigma,\sigma') \in \mathcal{C}[\![\mathbf{c_1}]\!]\}$$

$c ::=$ skip $\mid X := a \mid c_0; c_1 \mid$ if b then c_0 else c_1 $\mid$ while b do c

Denotational Semantics of **Com**

Definition

$$\mathcal{C}[\![\textbf{while } b \textbf{ do } c]\!] = \mathrm{fix}(\Gamma)$$

where

$$\Gamma(\varphi) = \{(\sigma, \sigma') \mid \mathcal{B}[\![b]\!]\sigma = \textbf{true} \ \& \ (\sigma, \sigma') \in \varphi \circ \mathcal{C}[\![c]\!]\}$$
$$\cup \{(\sigma, \sigma) \mid \mathcal{B}[\![b]\!]\sigma = \textbf{false}\}$$

$c ::=$ skip $\mid X := a \mid c_0; c_1 \mid$ if b then c_0 else c_1 $\mid$ while b do c

Denotational Semantics of **Com**

Theorem

Let $\Gamma(\varphi) = \{(\sigma, \sigma') \mid \mathcal{B}[\![b]\!]\sigma = \mathbf{true} \;\&\; (\sigma, \sigma') \in \varphi \circ \mathcal{C}[\![c]\!]\}$
$\qquad \cup \{(\sigma, \sigma) \mid \mathcal{B}[\![b]\!]\sigma = \mathbf{false}\}$

We define θ_n *as follows:*

$$
\begin{aligned}
\theta_0 &= \emptyset \\
\theta_{n+1} &= \{(\sigma, \sigma') \mid \mathcal{B}[\![b]\!]\sigma = \mathbf{true} \;\&\; (\sigma, \sigma') \in \theta_n \circ \mathcal{C}[\![c]\!]\} \\
&\quad \cup \{(\sigma, \sigma) \mid \mathcal{B}[\![b]\!]\sigma = \mathbf{false}\}.
\end{aligned}
$$

Then

$$\mathrm{fix}(\Gamma) = \cup_{n\in\omega}\theta_n.$$

Denotational Semantics of **Com**

Proposition

$$\mathcal{C}[\![\mathbf{skip}]\!]\sigma = \sigma$$

$$\mathcal{C}[\![X := a]\!]\sigma = \sigma[\mathcal{A}[\![a]\!]\sigma/X]$$

$$\mathcal{C}[\![c_0; c_1]\!]\sigma = \mathcal{C}[\![c_1]\!]\mathcal{C}[\![c_0]\!]\sigma$$

$$\mathcal{C}[\![\mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1]\!]\sigma = \begin{cases} \mathcal{C}[\![c_0]\!]\sigma & \text{if } \mathcal{B}[\![\mathbf{b}]\!]\sigma = \mathbf{true} \\ \mathcal{C}[\![c_1]\!]\sigma & \text{if } \mathcal{B}[\![\mathbf{b}]\!]\sigma = \mathbf{false} \end{cases}$$

c ::= skip | $X := a$ | $c_0; c_1$ | if b then c_0 else c_1 | while b do c

Denotational Semantics of **Com**

Proposition

$$\mathcal{C}[\![\textbf{while } b \textbf{ do } c]\!]\sigma = \mathrm{fix}(\Gamma)\sigma$$

where

$$\Gamma(\varphi)\sigma = \begin{cases} \varphi(\mathcal{C}[\![c]\!]\sigma) & \mathcal{B}[\![b]\!]\sigma = \textbf{true} \\ \sigma & \mathcal{B}[\![b]\!]\sigma = \textbf{false} \end{cases}$$

$\Gamma : (\Sigma \to \Sigma) \longrightarrow (\Sigma \to \Sigma)$
$\varphi : \Sigma \to \Sigma$.

$$c ::= \mathsf{skip} \mid X := a \mid c_0; c_1 \mid \mathsf{if}\ b\ \mathsf{then}\ c_0\ \mathsf{else}\ c_1 \mid \mathsf{while}\ b\ \mathsf{do}\ c$$

Denotational Semantics of **Com**

Proposition

Let $w \equiv$ while b do c. *Then*

$$\mathcal{C}[\![w]\!] = \mathcal{C}[\![\textbf{if } b \textbf{ then } c; w \textbf{ else skip}]\!].$$

Proof: The denotation of w is a fixed point of Γ. Hence

$$\begin{aligned}
\mathcal{C}[\![w]\!] &= \Gamma(\mathcal{C}[\![w]\!]) \\
&= \{(\sigma, \sigma') \mid \mathcal{B}[\![b]\!]\sigma = \textbf{true} \ \& \ (\sigma, \sigma') \in \mathcal{C}[\![w]\!] \circ \mathcal{C}[\![c]\!]\} \cup \\
&\quad\ \{(\sigma, \sigma) \mid \mathcal{B}[\![b]\!]\sigma = \textbf{false}\} \\
&= \{(\sigma, \sigma') \mid \mathcal{B}[\![b]\!] = \textbf{true} \ \& \ (\sigma, \sigma') \in \mathcal{C}[\![c; w]\!]\} \cup \\
&\quad\ \{(\sigma, \sigma') \mid \mathcal{B}[\![b]\!]\sigma = \textbf{false} \ \& \ (\sigma, \sigma) \in \mathcal{C}[\![\textbf{skip}]\!]\} \\
&= \mathcal{C}[\![\textbf{if } b \textbf{ then } c; w \textbf{ else skip}]\!].
\end{aligned}$$

Equivalence of the Semantics

Lemma

For all $a \in \mathbf{Aexp}$*,*

$$\mathcal{A}[\![a]\!] = \{(\sigma, n) \mid < a, \sigma > \rightarrow n\}.$$

Lemma

For all $b \in \mathbf{Bexp}$*,*

$$\mathcal{B}[\![b]\!] = \{(\sigma, t) \mid < b, \sigma > \rightarrow t\}.$$

Lemma

For all commands c *and states* σ, σ'*,*

$$< c, \sigma > \rightarrow \sigma' \text{ implies } (\sigma, \sigma') \in \mathcal{C}[\![c]\!], \text{ i.e., } \mathcal{C}[\![c]\!]\sigma = \sigma'.$$

Equivalence of the Semantics

Theorem

For all commands $c \in \mathbf{Com}$

$$\mathcal{C}[\![c]\!] = \{(\sigma, \sigma') \mid < c, \sigma > \rightarrow \sigma'\}.$$

That is: $\mathcal{C}[\![c]\!]\sigma = \sigma'$ *if and only if* $< c, \sigma > \rightarrow \sigma'$.

Proof: We prove the theorem by structural induction with a use of mathematical induction inside one case that for while-loops.

例子

(1) 设程序c_1为$X := (X+1)\times(Y+Z); Y := (X\times X+Y\times Y+2\times X\times Y); Z := X+Y-X\times Y$. 再设状态$\sigma$为: $\sigma(X)=8, \sigma(Y)=-2, \sigma(Z)=3$, 计算$\mathcal{C}[\![c_1]\!]\sigma$。

(2) 设程序c_2为$X := X\times Y;$ **if** $X \leq Y$ **then** $Y := (Y-X)\times(Z\times Y-X)$ **else** $Z := Z\times Y+X$, 计算程序c_2在下面状态σ处的指称语义值$\mathcal{C}[\![c_2]\!]\sigma$:

1. $\sigma_1(X)=6, \sigma_1(Y)=100, \sigma_1(Z)=-10$
2. $\sigma_2(X)=100, \sigma_2(Y)=10, \sigma_2(Z)=-10$
3. $\sigma_3(X)=100, \sigma_3(Y)=100, \sigma_3(Z)=100$

例子

(3) 设程序c_3为$X := X + Y - Z;$ **if** $X + 1 \leq X \times Y$ **then** $X := (X + 1) \times Z; Y := (X + Y) \times (Z - 1)$ **else** $(Z := X \times Y - 1; X := Y \times Z);$ **while** $X \leq 1000$ **do** $(Y := X + Y, Z := X \times Z; X := X + 2); X := Y; Y := Z; Z := X$, 设状态$\sigma$为: $\sigma(X) = 2, \sigma(Y) = 10, \sigma(Z) = 6$,计算程序$c_3$在此状态处的指称语义$\mathcal{C}[\![c_3]\!]\sigma$。

Axiomatic Semantics of **IMP**

1. **IMP** Language
2. Operational
1. Denotational
2. Axiomatic
3. UTP

Axiomatic Semantics of **IMP**

- Systematic verification of programs in **IMP**
- The Hoare rules: showing the partial correctness of programs
- Extending the boolean expressions to a rich language of assertions about program states

The Idea

- We consider the problem of how to prove that a program we have written in **IMP** does what we require of it.
- Simple example of a program to compute the sum of the first hundred numbers.
- A program in **IMP** to compute $\sum_{1 \leq m \leq 100} m$.

$$S := 0;$$

$$N := 1;$$

$$(\mathbf{while}\ \neg(N = 101)\ \mathbf{do}\ S := S + N; N := N + 1)$$

- How would we prove that this program , when it terminates, is such that the value of S is $\sum_{1 \leq m \leq 100} m$?

The Idea

$$S := 0; N := 1;$$

$$\{S = 0 \wedge N = 1\}$$

$$(\textbf{while}\ \neg(N = 101)\ \textbf{do}\ S := S + N; N := N + 1)$$

$$\{S = \sum_{1 \leq m \leq 100} m\}$$

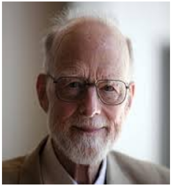
- Precondition: $S = 0 \wedge N = 1$
- Postcondition: $S = \sum_{1 \leq m \leq 100} m$

Floyd-Hoare Triple

- A proof system on assertion of the form

$$\{A\}c\{B\}$$

where A and B are assertions and c is a command. A is precondition and B is a postcondition.



Floyd-Hoare Triple

- The precise interpretation of such a compound assertion $\{A\}c\{B\}$ is this:
  For all sates σ which satisfy A if the execution c from state σ terminates in state σ' then σ' satisfies B.
- $\sigma \models A$ means that the assertion A is true at state σ.

$$\forall \sigma.(\sigma \models A \ \& \ \mathcal{C}[\![c]\!]\sigma \text{ is defined }) \text{ implies that } \mathcal{C}[\![c]\!]\sigma \models B.)$$

Floyd-Hoare Triple

- Partial correctness: $\{A\}c\{B\}$ is called partial correctness assertion because they say nothing about the command c if it fails to terminate.

$$\forall \sigma.(\sigma \models A \ \& \ \mathcal{C}[\![c]\!]\sigma \text{ is defined) implies that } \mathcal{C}[\![c]\!]\sigma \models B.)$$

- Total correctness: $\{A\}c\{B\}$ states that c terminates at state σ.

$$\forall \sigma.(\sigma \models A \text{ implies that } c \text{ terminates at state } \sigma \text{ and } \mathcal{C}[\![c]\!]\sigma \models B.)$$

The Assertion Language Assn

- The Arithmetic expressions **Aexpv**:

$$a := n \mid X \mid i \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$$

where

  - n ranges over numbers **N**
  - X ranges over locations **Loc**
  - i ranges over integer variables **Intvar**.

- The boolean assertions **Assn**

$$\begin{aligned} A \quad := \quad & \mathbf{true} \mid \mathbf{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \\ & A_0 \wedge A_1 \mid A_0 \vee A_1 \mid \neg A \mid A_0 \Rightarrow A_1 \mid \forall i.A \mid \exists i.A \end{aligned}$$

Semantics of Assertions

Definition

An interpretation is a function which assigns an integer to each integer variable, $i.e.$, a function

$$I : \mathbf{Intvar} \rightarrow \mathbf{N}.$$

Semantics of Assertions

Definition

Define the meaning of expression $a \in \mathbf{Aexpv}$ *in an interpretation* I *and a state* σ *by structural induction, as the value, , denoted by* $\mathcal{A}v[\![a]\!]I\sigma$.

$$\mathcal{A}v[\![n]\!]I\sigma = n$$

$$\mathcal{A}v[\![X]\!]I\sigma = \sigma(X)$$

$$\mathcal{A}v[\![i]\!]I\sigma = I(i)$$

$$\mathcal{A}v[\![a_0 + a_1]\!]I\sigma = \mathcal{A}v[\![a_0]\!] + \mathcal{A}v[\![a_1]\!]I\sigma$$

$$\mathcal{A}v[\![a_0 - a_1]\!]I\sigma = \mathcal{A}v[\![a_0]\!] - \mathcal{A}v[\![a_1]\!]I\sigma$$

$$\mathcal{A}v[\![a_0 \times a_1]\!]I\sigma = \mathcal{A}v[\![a_0]\!] \times \mathcal{A}v[\![a_1]\!]I\sigma$$

Semantics of Assertions

Proposition

For all $a \in \mathbf{Aexp}$ *(without integer variables), for all states* σ *and for all interpretations* I*,*

$$\mathcal{A}[\![a]\!]\sigma = \mathcal{A}v[\![a]\!]I\sigma.$$

Semantics of Assertions

Notations:

- $I[n/i]$ to mean the interpretation got from interpretation I by changing the value for inter-variable i to n, i.e.,

$$I[n/i](j) = \begin{cases} n & \text{if } j \equiv i \\ I(j) & \text{otherwise} \end{cases}$$

- $\Sigma_\perp = \Sigma \cup \{\perp\}$
- $\sigma \models^I A$ means state σ satisfies A in interpretation I, or equivalently, that assertion A is true at state σ.
- $\perp \models^I A$ is required.

Satisfaction relation

Definition

We define relation $\sigma \models^I A$ *for all* $\sigma \in \Sigma$ *and* $A \in \mathbf{Assn}$:

$$
\begin{array}{ll}
\sigma \models^I \mathbf{true} \qquad \perp \models^I A & \\
\sigma \models^I (a_0 = a_1) & \text{if } \mathcal{A}v[\![a_0]\!]I\sigma = \mathcal{A}v[\![a_1]\!]I\sigma \\
\sigma \models^I (a_0 \leq a_1) & \text{if } \mathcal{A}v[\![a_0]\!]I\sigma \leq \mathcal{A}v[\![a_1]\!]I\sigma \\
\sigma \models^I A \wedge B & \text{if } \sigma \models^I A \text{ and } \sigma \models^I B \\
\sigma \models^I A \vee B & \text{if } \sigma \models^I A \text{ or } \sigma \models^I B \\
\sigma \models^I \neg A & \text{if } \sigma \not\models^I A \\
\sigma \models^I \forall i.A & \text{if } \sigma \models^{I[n/i]} A \text{ for all } n \in \mathbf{N} \\
\sigma \models^I \exists i.A & \text{if } \sigma \models^{I[n/i]} A \text{ for some } n \in \mathbf{N}
\end{array}
$$

Partial Correctness Assertion

Definition

A partial correctness assertion has the form

$$\{A\}c\{B\}$$

where $A, b \in \mathbf{Assn}$ *and* $c \in \mathbf{Com}$*.*

Definition

Let I *be an interpretation. Let* $\sigma \in \Sigma_\perp$*. The satisfaction relation between states and partial correctness assertion, with respect to* I*, is defined by*

$$\sigma \models^I \{A\}c\{B\} \text{ iff } (\sigma \models^I A \text{ implies } \mathcal{C}[\![c]\!]\sigma \models^I B).$$

The Validity of Partial Correctness Assertion

Definition

We say a partial correctness assertion $\{A\}c\{B\}$ *validity , denoted by*

$$\models \{A\}c\{B\}$$

if for all interpretations I *and all states* σ

$$\sigma \models^I \{A\}c\{B\}.$$

Examples:

$$\models \{S = 0 \wedge N = 1\}(\text{while } \neg(N = 101) \text{ do } S := S+N; N := N+1)$$
$$\{S = \sum_{1 \le m \le 100} \}$$

$$\models \{i < X\}X := X + 1\{i < X\}$$

Proof rules for Partial Correctness: Hoare logic

Skip $$\frac{}{\{A\}\mathbf{skip}\{A\}}$$

Assignments $$\frac{}{\{B[a/X]\}X := a\{B\}}$$

Sequencing $$\frac{\{A\}c_0\{C\} \quad \{C\}c_1\{B\}}{\{A\}\ c_0; c_1\ \{B\}}$$

Conditionals $$\frac{\{A \wedge b\}c_0\{B\} \quad \{A \wedge \neg b\}c_1\{B\}}{\{A\}\ \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1\ \{B\}}$$

while – loops $$\frac{\{A \wedge b\}c\{A\}}{\{A\}\ \mathbf{while}\ b\ \mathbf{do}\ c\{A \wedge \neg b\}}$$

Consequence $$\frac{\models (A \Rightarrow A') \quad \{A'\}c\{B'\} \quad \models (B' \Rightarrow B)}{\{A\}c\{B\}}$$

Proof rules for Partial Correctness: Hoare logic

- Hoare rules: proof systems
- Proofs: derivations
- Theorem: any conclusion of a derivation
- We write $\vdash \{A\}c\{B\}$ as a theorem $\{A\}c\{B\}$.

Proof rules for Partial Correctness: Hoare logic

Compute the postcondition from precondition:
Given commands
$c := (d := d + 2; y := y + d; d := d + 2; y := y + d)$ and
precondition $y = x^2 \wedge d = 2x - 1$.

$$\{y = x^2 \wedge d = 2x - 1\}\ d := d + 2; y := y + d; d := d + 2; y := y + d\ \{??\}$$

How to compute the postcondition:

$$\begin{array}{rcl}
\{y = x^2 \wedge d = 2x - 1\} & d := d + 2 & \{y = x^2 \wedge d = 2x + 1\} \\
\{y = x^2 \wedge d = 2x + 1\} & y := y + d & \{y = (x+1)^2 \wedge d = 2(x+1) - 1\} \\
\{y = (x+1)^2 \wedge d = 2(x+1) - 1\} & d := d + 2 & \{y = (x+1)^2 \wedge d = 2(x+1) + 1\} \\
\{y = (x+1)^2 \wedge d = 2(x+1) + 1\} & y := y + d & \{y = (x+2)^2 \wedge d = 2(x+2) - 1\}
\end{array}$$

Hoare logic: Example

We want to verify that the command

$$w \equiv (\text{while } x > 0 \text{ do } Y := X \times Y; X := X - 1)$$

computing the factorial function
$n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1$ with $0! = 1$.
We prove that

$$\{X = n \wedge n \geq 0 \wedge Y = 1\} w \{Y = n!\}.$$

We need to find out an invariant for w—the while-loops command.

$$I \equiv (Y \times X! = n! \wedge X \geq 0).$$

Hoare logic: Example

We show that $I \equiv (Y \times X! = n! \wedge X \geq 0)$ indeed is an invariant:

$$\{I \wedge X > 0\}Y := X \times Y; X := X - 1\{I\}.$$

From the rule for assignment, we have

$$\{I[(X-1)/X]\}X := X - 1\{I\}$$

where $I[(X-1)/X] \equiv (Y \times (X-1)! = n! \wedge (X-1) \geq 0)$.
Again by the assignment rule:

$$\{I[(X-1)/X](X \times Y)/Y]\}Y := X \times Y\{I[(X-1)/X]\}$$

where
$I[(X-1)/X](X \times Y)/Y] \equiv (X \times Y \times (X-1)! = n! \wedge (X-1) \geq 0)$.

Hoare logic: Example

Thus by the rule for sequencing,

$$\{X \times Y \times (X-1)! = n! \wedge (X-1) \geq 0\} Y := X \times Y; X := (X-1) \{I\}$$

Clearly

$$\begin{aligned} I \wedge X > 0 &\Rightarrow Y \times X! = n! \wedge X \geq 0 \wedge X > 0 \\ &\Rightarrow Y \times X! = n! \wedge X \geq 1 \\ &\Rightarrow X \times Y \times (X-1)! = n! \wedge (X-1) \geq 0. \end{aligned}$$

Thus by the consequence rule

$$\{I \wedge X > 0\} Y := X \times Y; X := (X-1) \{I\}$$

establishing that I is an invariant.

Hoare logic: Example

Now applying the rule for while-loops we obtain

$$\{I\}w\{I \wedge X \not> 0\}.$$

Clearly $(X = n) \wedge (n \geq 0) \wedge (Y = 1) \Rightarrow I$, and

$$\begin{aligned} I \wedge X \not> 0 \quad & \Rightarrow Y \times X! = n! X \geq 0 \wedge X \not> 0 \\ & \Rightarrow Y \times X! = n! \wedge X = 0 \\ & \Rightarrow Y \times 0! = Y = n! \end{aligned}$$

Thus by the consequence rule we conclude

$$\{(X = n) \wedge (Y = 1)\}w\{Y = n!\}.$$

Hoare logic: Sound Theorem

Lemma

(Lemma 6.8) *Let* $a, a_0 \in \mathbf{Aexpv}$. *Let* $X \in \mathbf{Loc}$. *Then for all interpretations* I *and states* σ,

$$\mathcal{A}v[\![a_0[a/X]]\!]I\sigma = \mathcal{A}v[\![a_0]\!]I\sigma[\mathcal{A}v[\![a]\!]I\sigma/X].$$

Lemma

(Lemma 6.9) *Let* I *be an interpretation. Let* $B \in \mathbf{Assn}, X \in \mathbf{Loc}$ *and* $a \in \mathbf{Aexp}$. *For all states* $\sigma \in \Sigma$ $\sigma \models^I B[a/X]$ *iff* $\sigma[\mathcal{A}[\![a]\!]\sigma/X] \models^I B$.

Hoare logic: Sound Theorem

Theorem

(Theorem 6.11) *Let* $\{A\}c\{B\}$ *be a partial correctness assertion. If* $\vdash \{A\}c\{B\}$ *then* $\models \{A\}c\{B\}$.

Proof: Showing each rule is sound implies that every theorem is valid. We use both inductions on structure of commands and on the length of derivation of theorem.

The proof of Sound Theorem

The case of $c = \mathbf{Skip}$: $\vdash \{A\}\mathbf{Skip}\{B\}$ implies that $B = A$. But, $\models \{A\}\mathbf{Skip}\{A\} = \{A\}\mathbf{Skip}\{B\}$.
The case of $c = X := a$: $\vdash \{A\}X := a\{B\}$ implies $A = B[a/X]$. By Lemma 6.9, we have $\sigma \models B[a/X]$ iff $\sigma[\mathcal{A}\sigma/X] \models^I B$. Thus,

$$\sigma \models^I B[a/X] \Rightarrow \mathcal{C}[\![X := a]\!]\sigma = \sigma[\mathcal{A}[\![a]\!]\sigma/X] \models^I B.$$

and hence, $\models \{B[a/X]\}X := a\{B\}$.

The proof of Sound Theorem

The case of $c = c_0; c_1$: Assume $\vdash \{A\}c_0; c_1\{B\}$. Then by the operational semantics, there exists $C \in \mathbf{Assn}$ such that $\vdash \{A\}c_0\{C\}$ and $\vdash \{C\}c_1\{B\}$. By the induction on the length of derivation, we have $\models \{A\}c_0\{C\}$ and $\models \{C\}c_1\{B\}$.
Let I be an interpretation and state $\sigma \in \Sigma_\perp$. Suppose $\sigma \models^I A$. Then $\mathcal{C}[\![c_0]\!] \models^I C$ and $\mathcal{C}[\![c_1]\!](\mathcal{C}[\![c_0]\!]\sigma) \models^I B$. Hence $\models \{A\}c_0; c_1\{B\}$.

The proof of Sound Theorem

The case of $c = \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1$. If $\vdash \{A\}\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1\{B\}$, then $\vdash \{A \wedge b\}c_0\{B\}$ and $\vdash \{A \wedge \neg b\}c_1\{B\}$. By the induction, we have

$$\models \{A \wedge b\}c_0\{B\} \;\; \models \{A \wedge \neq gb\}c_1\{B\}.$$

Let I be an interpretation and states $\sigma \in \Sigma_\perp$. Suppose $\sigma \models^I A$. Then either $\sigma \models^I b$ or $\sigma \models^I \neg b$. So, we have that $\sigma \models^I A \wedge b$ and then $\mathcal{C}[\![c_0]\!]\sigma \models^I B$ or $\sigma \models^I A \wedge \neg b$ and then $\mathcal{C}[\![c_1]\!]\sigma \models^I B$. This ensures $\models \{A\}\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1\{B\}$.

The proof of Sound Theorem

The case of $w = \textbf{while } b \textbf{ do } c$: Assume $\vdash \textbf{while } b \textbf{ do } c$. Then by operational semantics, $\vdash \{A \wedge b\}c\{A\}$ and $B = A \wedge \neg b$. By the induction, we have $\models \{A \wedge b\}c\{A\}$. Let I be an interpretation and state $\sigma \in \Sigma_\perp$. We need to prove that

$$\sigma \models^I A \text{ implies } \mathcal{C}[\![w]\!]\sigma \models^I A \wedge \neg b.$$

Recall $[\![w]\!] = \cup_{n \in \omega} \theta_n$ where

$$\begin{aligned}
\theta_0 &= \emptyset \\
\theta_{n+1} &= \Gamma(\theta_n) \\
&= \{(\sigma, \sigma') \mid \mathcal{B}[\![b]\!]\sigma = \textbf{true} \,\&(\sigma, \sigma') \in \theta_n \circ \mathcal{C}[\![c]\!]\} \cup \{(\sigma, \sigma) \mid \mathcal{B}
\end{aligned}$$

We shall show by mathematical induction that $P(n)$ holds where

The Proof of Sound Theorem

Base case $n = 0$. Then $\theta_0 = \emptyset$. So, $P(n)$ holds (?).
Induction Step: Assume the induction hypothesis $P(n)$ holds for $n \geq 0$ and attempt to prove $P(n+1)$. Suppose $(\sigma, \sigma') \in \theta_{n+1}$ and $\sigma \models^I A$. Either
(1) $\mathcal{B}[\![b]\!]\sigma = \mathbf{true}$ and $(\sigma, \sigma') \in \theta_n \circ \mathcal{C}[\![c]\!]$, or
(2) $\mathcal{B}[\![b]\!]\sigma = \mathbf{false}$ and $\sigma' = \sigma$.
We show that in either case that $\sigma' \models^I A \wedge \neg b$.
Assume (1). $\mathcal{B}[\![b]\!]\sigma = \mathbf{true}$ implies $\sigma \models^I b$ and hence $\sigma \models^I A \wedge b$.
Also $(\sigma, \sigma') \in \mathcal{C}[\![c]\!]$ and $(\sigma'', \sigma') \in \theta_n$ for some state σ''. We obtain $\sigma'' \models^I A$.. So, we have that $\sigma' \models^I A \wedge \neg b$.
Asumme (2).

Exercise

Exercise: 6.16 pp96 Using the Hoare rule prove

$$\{N = n \wedge M = m \wedge 1 \leq n \wedge 1 \leq m\}\mathbf{Euclid}\{X = \mathbf{gcd}(n, m)\}$$

where

$$\begin{aligned}\mathbf{Euclid} \equiv \mathbf{while} \quad & \neg(M = N)\ \mathbf{do} \\ & \mathbf{if}\ M \leq N \\ & \ \ \mathbf{then}\ N := N - M \\ & \ \ \mathbf{else}\ M := M - N.\end{aligned}$$

作业

(1) 设**IMP**程序c定义为:
if $M \leq N$ **then** $N := N - M; M := N \times M$ **else**
$M := M - N; N := M \times N$.
使用**IMP**的操作语义计算c在下面的状态σ下结果σ'

- $\sigma(N) = 8, \sigma(M) = 6$
- $\sigma(N) = 6, \sigma(M) = 8$
- $\sigma(N) = 8, \sigma(M) = 8$

作业

(2) 设IMP程序Euclid定义如下：

$$
\begin{aligned}
&\textbf{while}\quad \neg(M = N)\ \textbf{do}\\
&\qquad \textbf{if}\ M \leq N\\
&\qquad\quad \textbf{then}\ N := N - M\\
&\qquad\quad \textbf{else}\ M := M - N.
\end{aligned}
$$

再设$\sigma(N) = 8, \sigma(M) = 10$ 按照IMP的指称语义计算$\mathcal{C}[\![\textbf{Euclid}]\!]\sigma$

(3) 使用Hoare规则证明

$$\{N = n \wedge M = m \wedge 1 \leq n \wedge 1 \leq m\}\textbf{Euclid}^X\{X = \textbf{gcd}(n, m)\}$$

其中$\textbf{gcd}(n, m)$是n, m的最大公因子，而程序$\textbf{Euclid}^X$定义为

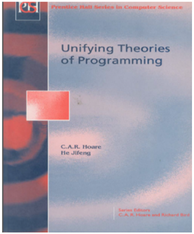
$$\left\{\begin{array}{ll} \textbf{while} & \neg(M = N)\ \textbf{do} \\ & \textbf{if}\ M \leq N \\ & \quad \textbf{then}\ N := N - M \\ & \quad \textbf{else}\ M := M - N \end{array}\right\}; X := N$$

Unifying Theories of Programming

1. IMP Language
2. Operational
1. Denotational
2. Axiomatic
3. UTP

Unifying Theories of Programming

Jim Woodcock said: The book by Hoare and He sets out a research programme to find a common basis in which to explain a wide variety of programming paradigms: unifying theories of programming (UTP). Their technique is to isolate important language features, and give them a denotational semantics. This allows different languages and paradigms to be compared.



Unifying Theories of Programming

- A simple language $\mathcal{H}$
- Denotational Semantics
- The laws
- Axiomatic Semantics
- Weakest precondition Semantics
- Dijkstra Healthiness Conditions

Unifying Theories of Programming–$\mathcal{H}$ Language

$\perp_A$	Abort
$\top_A$	Miracle
Π_A	Skip
$x := e$	Assignment
$P(v'); Q(v)$	Sequential composition
$P \lhd b \rhd Q$	Conditional
$P \sqcap Q$	Non-determinism
$P\|\|$	Parallel execution
$b * P$	While-loop

Denotational Semantics of $\mathcal{H}$ Language

Definition

(*Definition 2.0.1– Relation*)
A relation is a pair $(\alpha P, P)$, *where* P *is a predicate containing no free variables other than those in* αP, *and*

$$\alpha P = in\alpha P \cup out\alpha P.$$

Denotational Semantics of $\mathcal{H}$ Language

$$
\begin{aligned}
[\![\bot_A]\!] &= \textbf{true} & \alpha\bot_A &= A \\
[\![\top_A]\!] &= \textbf{false} & \alpha\top_A &= A \\
[\![\Pi_A]\!] &= (v' = v) & \alpha\Pi_A &= A = \{v, v'\} \\
[\![x := e]\!] &= (x' = e \wedge \cdots \wedge z' = z) & \alpha(x :=_A e) &= A \\
[\![P(v'); Q(v)]\!] &= \exists v_0 \bullet P(v_0) \wedge Q(v_0) & in\alpha(P(v'); Q(v)) &= in\alpha(P) \\
& & out\alpha(P(v'); Q(v)) &= out\alpha(Q) \\
[\![P \lhd b \rhd Q]\!] &= (b \wedge P) \vee (\neg b \wedge Q) & \alpha(P \lhd b \rhd Q) &= \alpha P \supseteq \alpha b \\
[\![P \sqcap Q]\!] &= P \vee Q & \alpha(P \sqcap Q) &= \alpha P = \alpha Q \\
[\![b * P]\!] &= \sqcap\{X \mid X \Rightarrow P; X \lhd b \rhd \Pi\} & \alpha(b * P) &= \alpha P \supseteq \alpha b
\end{aligned}
$$

Laws related to $\mathcal{H}$ Language

Proposition

(Assignment)

$$
\begin{array}{ll}
AL1 & (x := e) = (x, y := e, y) \\
AL2 & (x, y, z := e, f, g) = (y, x, z := f, e, g) \\
AL3 & (x := e; x := f(x)) = (x := f(e)) \\
AL4 & x := e; (P \lhd b(x) \rhd Q) = \\
 & (x := e; P) \lhd b(e) \rhd (x := e; Q)
\end{array}
$$

Laws related to $\mathcal{H}$ Language

Proposition

(Non-deterministic choice)

$$
\begin{array}{ll}
NDL1 & P \sqcap Q = Q \sqcap P \\
ND2 & P \sqcap (Q \sqcap R) = (P \sqcap Q) \sqcap R \\
NDL3 & P \sqcap P = P
\end{array}
$$

Laws related to $\mathcal{H}$ Language

Proposition

(Conditional)

$$
\begin{array}{ll}
CL1 & P \lhd b \rhd P = P \\
CL2 & P \lhd b \rhd Q = Q \lhd \neg b \rhd P \\
CL3 & (P \lhd b \rhd Q) \lhd c \rhd R = P \lhd b \wedge c \rhd (Q \lhd c \rhd R) \\
CL4 & P \lhd b \rhd (Q \lhd c \rhd R) = (P \lhd b \rhd Q) \lhd c \rhd (P \lhd b \rhd R) \\
CL5 & P \lhd true \rhd Q = P = Q \lhd false \rhd P
\end{array}
$$

Laws related to $\mathcal{H}$ Language

Proposition

(Sequencing Composition)

$$
\begin{array}{ll}
SL1 & P;(Q;R) = (P;Q);R \\
SL2 & (P \lhd b \rhd Q);R = (P;R) \lhd b \rhd (Q;R)
\end{array}
$$

Hoare triple of $\mathcal{H}$ Language

Definition

(Definition 2.8.1)

$$p\{Q\}r =_{def} [Q \Rightarrow (p \Rightarrow r')]$$

Hoare triple of $\mathcal{H}$ Language

(Theorem 2.8.2 –Hoare proof rules)

$HL1$ If $p\{Q\}r$ and $p\{Q\}s$ then $p\{Q\}(r \wedge s)$

$HL2$ If $p\{Q\}r$ and $q\{Q\}r$ then $p\{Q\}(r \vee s)$

$HL3$ If $p\{Q\}r$ then $(p \wedge q)\{Q\}(r \vee s)$

$HL4$ $r(e)\{x := e\}r(x)$

$HL5$ If $(p \wedge b)\{Q_1\}r$ and $(p \wedge \neg b)\{Q_2\}r$ then $p\{Q_1 \lhd b \rhd Q_2\}r$

$HL6$ If $p\{Q_1\}s$ and $s\{Q_2\}r$ then $p\{Q_1; Q_2\}r$

$HL7$ If $p\{Q_1\}r$ and $p\{Q_2\}r$ then $p\{Q_1 \sqcap Q_2\}r$

$HL8$ If $b \wedge c\{Q\}c$ them $c\{\nu X \bullet Q; X \lhd b \rhd \Pi\}(\neg b \wedge c)$

$HL9$ $\mathit{False}\{Q\}r$ and $p\{Q\}\mathit{true}$ and $p\{\mathsf{False}\}\mathit{false}$ and $p\{\Pi\}P$

Weakest precondition of $\mathcal{H}$ Language

Definition

(Definition 2.8.4)

$$
\begin{aligned}
Q\,\mathbf{wp}\,r &=_{def} \neg(Q; \neg r) \\
&= \neg(Q \wedge \neg r) \\
&= \neg Q \vee r
\end{aligned}
$$

Weakest precondition of $\mathcal{H}$ Language

Proposition

The laws related to weakest precondition

$$
\begin{array}{ll}
WL10 & (x := e) \ \mathbf{wp} \ r(x) = r(e) \\
WL11 & (P; Q) \ \mathbf{wp} \ r = P \ \mathbf{wp} \ (Q \ \mathbf{wp} \ r) \\
WL12 & (P \lhd b \rhd Q) \ \mathbf{wp} r = (P \ \mathbf{wp} \ \lhd b \ rhd(Q \ \mathbf{wp} \ r) \\
WL13 & (p \sqcap Q) \ \mathbf{wp} \ r = (P \ \mathbf{wp} \ r \wedge \ (Q \ \mathbf{wp} \ r)
\end{array}
$$

Healthiness conditions related to Weakest precondition

Proposition

$$
\begin{array}{ll}
DL14 & \text{If } [r \Rightarrow s] \text{ then } [Q \ \mathbf{wp}\ r \Rightarrow Q \ \mathbf{wp}\ s] \\
DL15 & \text{If } [Q \Rightarrow S] \text{ them } [S \ \mathbf{wp}\ r \Rightarrow Q \ \mathbf{wp}\ r] \\
DL16 & Q \ \mathbf{wp}\ (\wedge R) = \wedge\{Q \ \mathbf{wp}\ r \mid r \in R\} \\
DL17 & Q \ \mathbf{wp}\ \mathit{false} = \mathit{false}.
\end{array}
$$

Variable Declaration of $\mathcal{H}$ Language

Definition

(Definition 2.9.1)
Let A be an alphabet which includes x and x'. Then

$$\begin{array}{lllll} \mathsf{var}\ x & =_{def} & \exists x \bullet \Pi_A & \alpha(\mathsf{var}\ x) & =_{def} & A \setminus \{x\} \\ \mathsf{end}\ x' & =_{def} & \exists x' \bullet \Pi_A & \alpha(\mathsf{end}\ x') & =_{def} & A \setminus \{x'\} \end{array}$$

Definition

$$\begin{array}{rcl} \mathsf{var}\ x; Q & = & \exists x \bullet Q \\ Q; \mathsf{end}\ x & = & \exists x' \bullet Q \end{array}$$

Variable Declaration of $\mathcal{H}$ Language

Proposition

The Law related to Declaration

DL5 If x is not free in b, then

$$\text{var } x; (P \lhd b \rhd Q) = (\text{var } x : P) \lhd b \rhd (\text{var } x : Q)$$

$$\text{end } x; (P \lhd b \rhd Q) = (\text{end } x : P) \lhd b \rhd (\text{end } x : Q)$$

DL6 $\text{var } x; \text{end } x = \Pi$

DL7 $(\text{end } x;\ \text{var } x := e) - (x := e)$

DL8 $(x := e;\ \text{end } x) = \text{end } x$

The End

Email: yxchen@sei.ecnu.ecu.cn