



Floyd/Hoare Logic

Literature: peled ch. 7 – 7.5

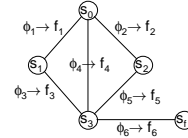
Mads Dam

Transition Diagrams

Transition system specs, with explicit underlying control graph

Labelled directed graph (S, Σ, R, s_0, s_f) :

- $s \in S$: Control states
- $\alpha = \phi \rightarrow (x_1, \dots, x_n) := (e_1, \dots, e_n) \in \Sigma$: Transition specification
- $R \subseteq S \times \Sigma \times S$: (Control) transition relation
- $s \rightarrow^\alpha s'$: Means $R(s, \alpha, s')$
- $s_0 \in S$: Initial state
- $s_f \in S$: Final state
- s_i should not have outgoing edges



Generated state space has states $(s, x_1=v_1, \dots, x_n=v_n)$
Stores σ range over data vectors $(x_1=v_1, \dots, x_n=v_n)$

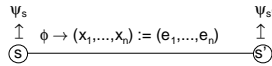
Floyd Inductive Assertions

Assume transition diagram $P = (S, \Sigma, R, s_i, s_f)$

Assertion network:

Assignment N: $s \mapsto \psi_s$ of total predicates to control states in S

N is *inductive* if whenever



then $\models \psi_s(\sigma)$ and $\models \phi(\sigma)$ then $\models \psi_{s'}[e_1/x_1, \dots, e_n/x_n](\sigma)$

Formally: $\models \psi_s \wedge \phi \rightarrow (\psi_{s'}[e_1/x_1, \dots, e_n/x_n])$

An assertion network N is *invariant* if for all computation paths

$$(s_0, \sigma_0) \rightarrow \dots \rightarrow (s_i, \sigma_i) \rightarrow \dots$$

such that $\models \psi_{s_0}(\sigma_0)$, also $\models \psi_{s_i}(\sigma_i)$, for any $i \geq 0$

An assertion network N is *consistent*, or *correct*, w.r.t. precondition

ϕ_{pre} and postcondition ϕ_{post} if:

- $\models \phi_{pre} \rightarrow \psi_{s_0}$, and
- $\models \psi_{s_f} \rightarrow \phi_{post}$

A transition diagram P is *partially correct* w.r.t. precondition ϕ_{pre} and postcondition ϕ_{post} if whenever $\models \phi_{pre}(\sigma_0)$ and

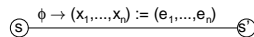
$$(s_0, \sigma_0) \rightarrow \dots \rightarrow (s_i, \sigma_i) \rightarrow \dots \rightarrow (s_f, \sigma_f)$$

then $\models \phi_{post}(\sigma_f)$

Partial correctness of P w.r.t. ϕ_{pre} and ϕ_{post} is written $\{ \phi_{pre} \} P \{ \phi_{post} \}$

Floyd's Inductive Assertion Method

1. Give assertion network N for P
2. Prove that N is inductive, i.e. prove that whenever



then $\models \psi_s \wedge \phi \rightarrow \psi_{s'}[e_1/x_1, \dots, e_n/x_n]$

3. Prove that N is consistent w.r.t. ϕ_{pre} and ϕ_{post} , i.e. that

- $\models \phi_{pre} \rightarrow \psi_{s_0}$
- $\models \psi_{s_f} \rightarrow \phi_{post}$

Then P is partially correct w.r.t. ϕ_{pre} and ϕ_{post}

Inductive Assertion Method: Soundness

Theorem:

If N is an inductive assertion network for P which is consistent w.r.t. ϕ_{pre} and ϕ_{post} then P is partially correct w.r.t. ϕ_{pre} and ϕ_{post}

Lemma:

If N is an inductive assertion network for P then N is invariant for P

Proof: Induction on length of prefix $(s_0, \sigma_0) \rightarrow \dots \rightarrow (s_i, \sigma_i)$

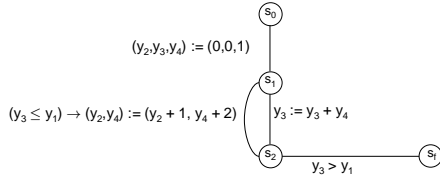
Lemma:

If N is invariant for P and consistent w.r.t. ϕ_{pre} and ϕ_{post} then $\{ \phi_{pre} \} P \{ \phi_{post} \}$

Example

Procedure for computing integer square root of nonnegative integer y_1 , with result in y_2

Integer square root: y_2 s.t. $y_2^2 \leq y_1 < (y_2+1)^2$

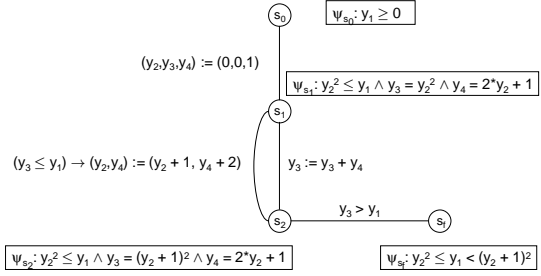


2005 Mads Dam IMIT, KTH

7

2G1516 Formal Methods

Example



2005 Mads Dam IMIT, KTH

8

2G1516 Formal Methods

Semantic Completeness

Soundness: Whenever $\{\phi_{pre}\} P \{\phi_{post}\}$ is proved using the inductive assertion method then $\{\phi_{pre}\} P \{\phi_{post}\}$ is valid
Completeness: The inductive assertion method is sufficient to derive any valid partial correctness property $\{\phi_{pre}\} P \{\phi_{post}\}$

For completeness prove the existence of network N such that $\models \phi_{pre} \rightarrow \psi_{N,s_0}$ and $\models \psi_{N,s_i} \rightarrow \phi_{post}$
 Obs: Doesn't prove that the ψ_s are expressible in any given logic

The derived ass'n network N is minimal in the sense that if M is some other ass'n network which establishes partial correctness of P w.r.t. ϕ_{pre} and ϕ_{post} then $\psi_{N,s} \rightarrow \psi_{M,s}$ for all $s \in S$

In other words, $\{\psi_{N,s} \mid s \in S\}$ is the set of strongest = least inclusive predicates such that $\{\phi_{pre}\} P \{\phi_{post}\}$

Notation: $\psi_{N,s} = SP_s(\phi_{pre}, P)$, $SP_{s_1}(\phi_{pre}, P) = SP(\phi_{pre}, P)$

2005 Mads Dam IMIT, KTH

9

2G1516 Formal Methods

Proof of Semantic Completeness

Suppose $\{\phi_{pre}\} P \{\phi_{post}\}$

Define:

$$SP_s(\phi_{pre}, P) = \{\sigma' \mid \exists \sigma. (s_0, \sigma) \rightarrow^* (s, \sigma') \text{ and } \models \phi_{pre}(\sigma)\}$$

The assertion network N determined by

$$\psi_s = SP_s(\phi_{pre}, P)$$

is inductive:

- If $\models \psi_s(\sigma)$, $s \rightarrow^{s'} s'$, and $\models \phi(s)$ then $\models \psi_{s'}(\sigma')$

N is also consistent w.r.t. ϕ_{pre} and ϕ_{post} :

- $SP_{s_1}(\phi_{pre}, P) = \phi_{pre}$, so $\models \phi_{pre} \rightarrow \psi_{s_0}$
 - N is inductive, hence invariant. We assumed $\{\phi_{pre}\} P \{\phi_{post}\}$. But then $\models SP_{s_1}(\phi_{pre}, P) \rightarrow \phi_{post}$

Since N is inductive and consistent w.r.t. ϕ_{pre} and ϕ_{post} the inductive assertions method applies

2005 Mads Dam IMIT, KTH

10

2G1516 Formal Methods

Strongest Postconditions

$$\begin{aligned} SP(\phi, P) &= SP_{s_1}(\phi, P) \\ &= \{\sigma' \mid \exists \sigma. (s_0, \sigma) \rightarrow^* (s_1, \sigma') \text{ and } \models \phi(\sigma)\} \end{aligned}$$

Lemma:

- $\models \{\phi\} P \{SP(\phi, P)\}$
- If $\models \{\phi\} P \{\psi\}$ then $\models SP(\phi, P) \rightarrow \psi$

2. explains why $SP(\phi, P)$ is called strongest

2005 Mads Dam IMIT, KTH

11

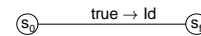
2G1516 Formal Methods

Incompleteness

By Gödel's incompleteness theorem no complete proof system can exist for FOL + (Peano) arithmetic

It follows that the inductive assertion method is incomplete too:

Consider P:



with specification $\{\text{true}\} P \{\psi\}$ such that $\models \psi$

Completeness would require us to prove ψ which is not generally possible

2005 Mads Dam IMIT, KTH

12

2G1516 Formal Methods

Total Correctness

Total correctness = partial correctness + termination

- This terminology is from the days when programs were by default sequential and terminating

A transition diagram P is *totally correct* w.r.t. precondition ϕ_{pre} and postcondition ϕ_{post} if whenever $\models \phi_{pre}(\sigma_0)$ and $(s_0, \sigma_0) \rightarrow \dots \rightarrow (s_i, \sigma_i) \rightarrow \dots$ is maximal then $s_i = s_f$ for some i , and $\models \phi_{post}(\sigma_i)$

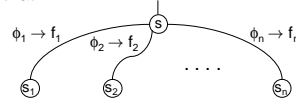
Termination is about *progressing* towards a terminal state

So is *induction*

For termination proofs need general induction principle called *well-founded induction*, but here ordinary induction suffices

Deadlock-free Networks

To avoid states (s, σ) such that $(s, \sigma) \nrightarrow$ but $s \neq s_f$ we assume that if



are all control transitions emanating from control state $s \neq s_f$ then

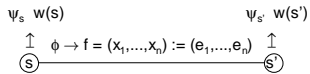
$$\models \phi_1 \vee \phi_2 \vee \dots \vee \phi_n$$

Extended Inductive Assertions

Extended assertion network:

In addition to assertion network N :

Associate to each control state s a natural number $w(s)$ s.t. whenever



then

1. $\models \psi_s \rightarrow w(s) \in W$
2. $\models \psi_s \wedge \phi \rightarrow w(s) \geq w(s')[e_i/x_i, \dots, e_n/x_n]$
3. For each cycle (= strongly connected subset) there is at least one transition as above such that $\models \psi_s \wedge \phi \rightarrow w(s) > w(s)[e_i/x_i, \dots, e_n/x_n]$

Say N is progressing if an assignment w satisfying 1.-3. exists

Extended Inductive Assertion Method

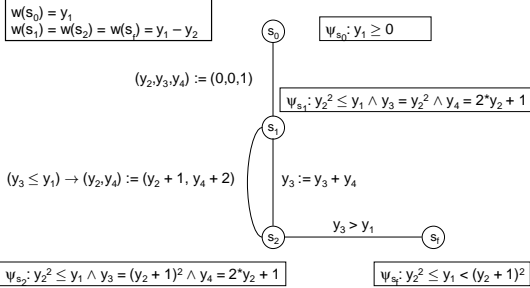
1. Give assertion network N for P
2. Prove that the network is inductive
3. Prove that N is consistent w.r.t. ϕ_{pre} and ϕ_{post}
4. Prove that N is deadlock-free
5. Determine assignment w
6. Prove that N with this assignment is progressing

Then P is totally correct w.r.t. ϕ_{pre} and ϕ_{post}

Theorem

The extended inductive assertion method is sound

Example



While programs

Primitive:

- $x \in X$: set of identifiers
- $e \in E$: set of expressions
- $v \in V$: set of values

Command syntax in BNF:

$c ::= \text{skip} \mid x := e \mid c ; c \mid \text{if } e \text{ then } c \text{ else } c \mid \text{while } e \text{ do } c$

Exercise: Cast the command syntax as first-order structure

$\Sigma = \{.\}$ (will remain so for a while)

Stores

Stores are assignments $\sigma: x \mapsto v$ of values to identifiers

$e(\sigma)$: value of e in store σ

Store update:

$\sigma[x \mapsto v](y) = \text{if } x=y \text{ then } v \text{ else } \sigma(y)$

States are either

- Intermediate: Pairs of commands and stores (c, σ) , or
- Final: A state σ

While Programs

Transitions inductively defined by inference system:

$$\frac{}{(\text{skip}, \sigma) \rightarrow \sigma} \quad \frac{}{(x:=e, \sigma) \rightarrow \sigma[x \mapsto e(\sigma)]}$$

$$\frac{(c_1, \sigma) \rightarrow \sigma'}{(c_1; c_2, \sigma) \rightarrow (c_2, \sigma')} \quad \frac{(c_1, \sigma) \rightarrow (c_1', \sigma')}{(c_1, c_2, \sigma) \rightarrow (c_1'; c_2, \sigma')}$$

$$\frac{e(\sigma) \neq 0}{(\text{if } e \text{ then } c_1 \text{ else } c_2, \sigma) \rightarrow (c_1, \sigma)}$$

$$\frac{e(\sigma) = 0}{(\text{if } e \text{ then } c_1 \text{ else } c_2, \sigma) \rightarrow (c_2, \sigma)}$$

While Programs, II

$$\frac{e(\sigma) \neq 0}{(\text{while } e \text{ do } c, \sigma) \rightarrow (c; \text{while } e \text{ do } c, \sigma)}$$

$$\frac{e(\sigma) = 0}{(\text{while } e \text{ do } c, \sigma) \rightarrow \sigma}$$

Exercise: Let $c_1 = x:=1; \text{while } x>0 \text{ do } x:=x-1$. Pick an arbitrary σ_1 . Compute a sequence $(c_1, \sigma_1) \rightarrow (c_2, \sigma_2) \rightarrow \dots \rightarrow \sigma_n$

Exercise: Prove that \rightarrow is *deterministic*, i.e. that for any c, σ there is at most one c', σ' such that $(c, \sigma) \rightarrow (c', \sigma')$

Exercise (more advanced): Try to add some new language construction, like choice, cobegin/coend, or variable declarations. Add new components to the state if you want.

Hoare Logic

Hoare triple $\{\phi\} c \{\psi\}$:

- Starting in state satisfying ϕ , if and when c terminates, ψ holds
- Or: Whenever $\models \phi(\sigma)$ and $(c, \sigma) = (c_0, \sigma_0) \rightarrow (c_1, \sigma_1) \rightarrow \dots \rightarrow \sigma_1$ then $\models \psi(\sigma_1)$
- I.e. c is partially correct w.r.t. ϕ and ψ

Inference Rules

Assignment:

$$\frac{}{\{\phi[e/v]\} v := e \{\phi\}}$$

Skip:

$$\frac{}{\{\phi\} \text{skip} \{\phi\}}$$

Rule of consequence:

$$\frac{\models \phi \rightarrow \phi' \quad \{\phi'\} c \{\psi'\} \quad \models \psi' \rightarrow \psi}{\{\phi\} c \{\psi\}}$$

Inference Rules, II

Sequential composition

$$\frac{\{\phi\} c_1 \{\psi\} \quad \{\psi\} c_2 \{\gamma\}}{\{\phi\} c_1; c_2 \{\gamma\}}$$

Conditional

$$\frac{\{\phi \wedge e \neq 0\} c_1 \{\psi\} \quad \{\phi \wedge e = 0\} c_2 \{\psi\}}{\{\phi\} \text{if } e \text{ then } c_1 \text{ else } c_2 \{\psi\}}$$

While

$$\frac{\{\phi \wedge e \neq 0\} c \{\phi\}}{\{\phi\} \text{while } e \text{ do } c \text{ od } \{\phi \wedge e = 0\}}$$

Example

The integer square root example again:

```
P:  y2 := 0 ;
    y3 := 1 ;
    y4 := 1 ;
    while y3 <= y1 do
      y2 := y2 + 1 ;
      y4 := y4 + 2 ;
      y3 := y3 + y4
    od
```

Proof goal: $\{y1 \geq 0\} P \{y2^2 \leq y1 < (y2 + 1)^2\}$

Proof Outlines

State predicates inserted into program text such that each statement (simple or compound) has pre- and postcondition

Proof outline is valid, if each embedded triple is valid and adjacent state predicates related by implication

Proof Outlines, Example

```
P: {y1>=0}
   y2 := 0 ;
   {y1>=0 ∧ y2=0}
   y3 := 1 ;
   {y1>=0 ∧ y2=0 ∧ y3=1}
   y4 := 1 ;
   {y1>=0 ∧ y2=0 ∧ y3=1 ∧ y4=1}
   {y2^2<=y1 ∧ y3=(y2+1)^2 ∧ y4=2*y2+1}
   while y3 <= y1 do
     {y2^2<=y1 ∧ y3=(y2+1)^2 ∧ y4=2*y2+1 ∧ y3<=y1}
     y2 := y2 + 1 ;
     {y2^2<=y1 ∧ y3=y2^2 ∧ y4=2*y2-1}
     y4 := y4 + 2 ;
     {y2^2<=y1 ∧ y3=y2^2 ∧ y4=2*y2+1}
     y3 := y3 + y4
     {y2^2<=y1 ∧ y3=(y2+1)^2 ∧ y4=2*y2+1}
   od
   {y2^2 <= y1 < (y2+1)^2} /* Postcondition */
```

Soundness and Completeness

Theorem (soundness):

If $\{\phi\} c \{\psi\}$ is provable then c is partially correct w.r.t. ϕ and ψ

For the case of sequential composition and while, let

$(c, \sigma) \rightarrow^n \sigma'$ if $(c, \sigma) \rightarrow \dots \rightarrow \sigma'$ in "n steps"

Lemma: If $(c_1; c_2, \sigma) \rightarrow^n \sigma'$ then there are n_1, n_2, σ'' such that $(c_1, \sigma) \rightarrow^{n_1} \sigma''$, $(c_2, \sigma'') \rightarrow^{n_2} \sigma'$ and $n = n_1 + n_2$

Completeness:

Can obtain relative completeness, completeness relative to oracle answering true statements in FOL + arithmetic