CS256/Spring 2008 — Lecture #7

Zohar Manna

Strengthening vs. Incremental Proof

Comparing the Strategies

We want to prove $\Box q$, but q is not inductive.

We have two options:

[1] Strengthening
Strengthen it to $q \wedge \varphi$.
Prove $\Box(q \wedge \varphi)$ and deduce $\Box q$.

[2] Incremental
First prove $\Box \varphi$ and then prove $\Box q$ relative to φ.

Resulting verification conditions:

[1]
I1. $\Theta \rightarrow q \wedge \varphi$
I2. $\{q \wedge \varphi\}\ \mathcal{T}\ \{q \wedge \varphi\}$

[2]

$$\frac{\text{I1'}.\ \Theta \rightarrow \varphi \qquad \text{I2'}.\ \{\varphi\}\ \mathcal{T}\ \{\varphi\}}{\Box \varphi} \qquad \frac{\text{I1''}.\ \Theta \rightarrow q \qquad \text{I2''}.\ \{q \wedge \varphi\}\ \mathcal{T}\ \{q\}}{\Box q}$$

Strengthening vs. Incremental Proof (Con't)

- [1] is strictly more powerful than [2].
  [2] implies [1] since

$$\begin{bmatrix} \underbrace{\rho_\tau \wedge \varphi \ \rightarrow \varphi'}_{\text{I2'}} \\ \\ \underbrace{\rho_\tau \wedge q \wedge \varphi \rightarrow q'}_{\text{I2''}} \end{bmatrix} \rightarrow \underbrace{[\rho_\tau \wedge q \wedge \varphi \ \rightarrow \ q' \wedge \varphi']}_{\text{I2}}$$

- In practice, [2] is often more useful than [1]
  - allows breaking down the proof in more manageable pieces
  - smaller verification conditions
  - more intuitive

Strengthening vs. Incremental Proof (Con't)

`Example:`

local x: **integer where** $x = 1$
ℓ_0: **loop forever do**
$\quad [\ \ell_1 :\ x := x + 1\ ]$

Show q_1: $at\_\ell_0 \rightarrow x > 0$

q_2: $at\_\ell_1 \rightarrow x > 0$

- both are P-valid
- neither of them is inductive
- but $q_1 \wedge q_2$ is inductive!

Combining the Strategies

Rule INC-INV: (incremental invariance)

For assertions $q,\varphi,\chi_1,\ldots,\chi_k$

I0. $P \vDash \Box\,\chi_1,\ldots,\Box\,\chi_k$

I1. $P \Vdash (\bigwedge_{i=1}^{k} \chi_i) \wedge \varphi \rightarrow q$

I2. $P \Vdash \Theta \rightarrow \varphi$

I3. $P \Vdash \left\{(\bigwedge_{i=1}^{k} \chi_i) \wedge \varphi\right\} \mathcal{T} \{\varphi\}$

$P \vDash \Box\, q$

If φ satisfies I2 and I3, we say that
"φ is inductive relative to $\chi_1,\ldots\chi_k$"

Combining the Strategies (Con't)

Note that Θ must be stronger than all the χ_i's (i.e., $P \Vdash \Theta \rightarrow \chi_i$) and so

$$P \Vdash \left(\bigwedge_{i=1}^{k} \chi_i\right) \wedge \Theta \rightarrow \varphi \quad \text{iff} \quad P \Vdash \Theta \rightarrow \varphi$$

From now on, we usually omit "$P \vDash$" and "$P \Vdash$".

Detecting Trivial Verification Conditions

$\{\varphi\}\ \mathcal{T}\ \{\varphi\}$ – Don't check every $\tau \in \mathcal{T}$.

- Ignore $\{\varphi\}\ \tau_I\ \{\varphi\}$ – always true
- Ignore $\{\varphi\}\ \tau\ \{\varphi\}$
  if τ does not modify any variable in φ
- For $\{\varphi\}\ \tau\ \{\varphi\}$ where φ: $p \rightarrow q$

$$\rho_\tau \wedge \underbrace{p \rightarrow q}_{\varphi} \rightarrow \underbrace{p' \rightarrow q'}_{\varphi'}$$

Consider only τ's that
validate p or falsify q

Finding Inductive Assertions

Two methods:

1. Bottom-up:
   - based on the program text only
   - algorithmic
   - guaranteed to produce an inductive invariant
2. Top-down:
   - guided by the property we want to prove
   - heuristic
   - not guaranteed to produce an inductive invariant

Finding Inductive Assertions
Bottom-Up Approach

- Transition-validated assertions:

ℓ_1: [**while** c **do** S]; ℓ_2: $\boxed{at\_\ell_2 \;\rightarrow\; \neg c}$

if no statement parallel to ℓ_2 can modify variables in c

ℓ_1: $y := e$; ℓ_2: $\boxed{at\_\ell_2 \;\rightarrow\; y = e}$

if no statement parallel to ℓ_2 can modify y or variables occurring in e and if y does not occur in e.

Bottom-Up Approach (Con't)

- single variable assertions

$$y = 1$$

$$\left[\begin{array}{l}\textbf{loop forever do}\\ \quad\left[\begin{array}{l}\cdots\\ \textbf{request } y\\ \cdots\\ \textbf{release } y\end{array}\right]\end{array}\right] \qquad \boxed{y \geq 0}$$

$$s = 1$$

$$\left[\begin{array}{l}\cdots\\ s := 1\\ \cdots\end{array}\right] \; || \; \left[\begin{array}{l}\cdots\\ s := 2\\ \cdots\end{array}\right] \qquad \boxed{s = 1 \;\vee\; s = 2}$$

where no other statement modifies s

`Example:` Program SQUARE-ROOT
Fig. 1.11

$\boxed{at\_\ell_2 \;\rightarrow\; z^2 \leq x < (z+1)^2}$

Intuitive argument:

$$z \;=\; 0, 1, \ldots, n$$
$$u \;=\; 1, 3, \ldots, 2n+1$$
$$w \;=\; \underbrace{1 + 3 + \ldots + (2n{+}1)}_{(n+1)^2} \;=\; (z+1)^2$$

first time $w > x$

$$x < (z+1)^2$$

last time $w \leq x$

$$z^2 \leq x$$

Thus at ℓ_2:

$$z^2 \leq x < (z+1)^2$$

Program SQUARE-ROOT

```
in     x:     integer  where x ≥ 0
local  u, w:  integer  where u = 1, w = 1
out    z:     integer  where z = 0

ℓ0 : while w ≤ x do
       ℓ1 :  (z, u, w) := (z + 1, u + 2, w + u + 2)
ℓ2 :
```

$$\rho_{\ell_0}:\quad \underbrace{move(\ell_0, \ell_1) \;\wedge\; w \leq x}_{\rho^{\mathrm{T}}_{\ell_0}} \;\;\vee$$
$$\underbrace{move(\ell_0, \ell_2) \;\wedge\; w > x}_{\rho^{\mathrm{F}}_{\ell_0}}$$

$$\rho_{\ell_1}:\quad \begin{array}{ll} move(\ell_1, \ell_0) & \wedge\\ z' = z + 1 & \wedge\\ u' = u + 2 & \wedge\\ w' = w + u + 2 & \end{array}$$

Find $\psi_2\colon\ at\_\ell_2 \ \rightarrow\ x < (z+1)^2$

$$\begin{cases} z_0 = 0 \\ z_n = z_{n-1} + 1 \quad \text{for } n > 0 \end{cases}$$

$$\begin{cases} u_0 = 1 \\ u_n = u_{n-1} + 2 \quad \text{for } n > 0 \end{cases}$$

$$\begin{cases} w_0 = 1 \\ w_n = w_{n-1} + u_{n-1} + 2 \quad \text{for } n > 0 \end{cases}$$

• Step 1

$$\left.\begin{array}{ll} z_n = n & \text{for } n \geq 0 \\ u_n = 2n+1 & \text{for } n \geq 0 \end{array}\right\} \Rightarrow \begin{array}{l} u_n = 2z_n + 1 \\ \text{for } n \geq 0 \end{array}$$

$\boxed{\varphi_1\colon\ u = 2z + 1}$

• Step 2

$$\begin{cases} w_0 = 1 \\ w_n = w_{n-1} + (\overbrace{2(n-1)+1}^{u_{n-1}}) + 2 \\ \quad = w_{n-1} + (2n+1) \qquad \text{for } n \geq 0 \end{cases}$$

$$w_n = \sum_{k=0}^{n} (2k+1) = (n+1)^2 \quad \text{for } n \geq 0$$

$w_n = (z_n + 1)^2 \quad \text{for } n \geq 0$

$\boxed{\varphi_2\colon\ w = (z+1)^2}$

• Step3

$\boxed{at\_\ell_2 \ \rightarrow\ x < w}$

Therefore

$\boxed{\psi_2\colon\ at\_\ell_2 \ \rightarrow\ x < (z+1)^2}$

Construction of Linear Invariants

a limited class of invariants that can be constructed algorithmically

Definition: integer variable y is linear in P if

$y' = y + c \qquad$ for every ρ_τ

for some integer constant c.

`Example:` semaphore variables are linear

$$\underbrace{y' = y + 1}_{\textbf{release}} \qquad \underbrace{y' = y - 1}_{\textbf{request}} \qquad \underbrace{y' = y}_{\text{otherwise}}$$

Definition:

A linear invariant is of the form

$$\boxed{\underbrace{\sum_{i=1}^{r} a_i \cdot y_i}_{\text{body}} + \underbrace{\sum_{\ell \in \mathcal{L}} b_\ell \cdot at\_\ell}_{\text{compensation expression}} = \underbrace{K}_{\text{constant}}}$$

where

$a_i,\ b_\ell,\ K$ – integer constants.

$\mathcal{L}$ – set of all locations in P

$y_1, \ldots, y_r$ – all linear variables in P

`Example:` Program DOUBLE

$$\begin{array}{c}\textbf{local } y\text{: } \textbf{integer where } y = 0 \\ \begin{bmatrix}\ell_0\text{: } y := y+1 \\ \ell_1\text{:}\end{bmatrix} \parallel \begin{bmatrix}m_0\text{: } y := y+1 \\ m_1\text{:}\end{bmatrix}\end{array}$$

linear variable: y

linear invariant:

$$\boxed{y + at\_\ell_0 + at\_m_0 \;=\; 2}$$

How are linear invariants constructed?

Our procedure guarantees that the generated assertions are P-invariants!

Assumption

Program $\ell_0^1\text{: } S_1 \parallel \ldots \parallel \ell_0^i\text{: } S_i \parallel \ldots \parallel \ell_0^m\text{: } S_m$

- no nested parallel statements. Therefore, all move expressions in all ρ_τ are of the form $move(\ell_i, \ell_j)$
- all linear variables y_i have a single initial value y_i^0
- every transition τ enabled on some P-accessible state

Increments

- $\Delta(y, \tau) = c$ if $\rho_\tau \rightarrow y' = y + c$
  therefore $\rho_\tau \rightarrow y' = y + \Delta(y, \tau)$
- $\Delta(at\_\ell, \tau) \;=\; \begin{cases} 1 & \text{if } \ell = \ell_j \\ -1 & \text{if } \ell = \ell_i \\ 0 & \text{otherwise}\end{cases}$
  if $\rho_\tau \rightarrow move(\ell_i, \ell_j)$
  therefore $\rho_\tau \rightarrow at'\_\ell = at\_\ell + \Delta(at\_\ell, \tau)$

Equations

Construct

$$\varphi\text{:}\quad \sum_{i=1}^{r} a_i \cdot y_i \;+\; \sum_{\ell \in \mathcal{L}} b_\ell \cdot at\_\ell \;=\; K$$

We obtain the values of the coefficients from a set of equations as follows:

(I) The invariant has to hold at the first state of every computation

$$\Theta \quad \text{implies} \quad y_i = y_i^0 \; (i = 1 \ldots r) \text{ and } \pi = \{\ell_0^1, \ldots, \ell_0^m\}$$

and so we get

$$\boxed{\sum_{i=1}^{r} a_i \cdot y_i^0 \;+\; (b_{\ell_0^1} + \cdots + b_{\ell_0^m}) \;=\; K}$$

Equations (Con'd)

(T) the assertion has to be preserved by all transitions (we want it to be inductive):

$$\underbrace{\left(\sum_{i=1}^{r} a_i \cdot y_i \;+\; \sum_{\ell \in \mathcal{L}} b_\ell \cdot at\_\ell = K\right)}_{\varphi} \wedge \rho_\tau \rightarrow \underbrace{\left(\sum_{i=1}^{r} a_i \cdot y_i' \;+\; \sum_{\ell \in \mathcal{L}} b_\ell \cdot at'\_\ell = K\right)}_{\varphi'}$$

or

$$\rho_\tau \;\rightarrow\; \sum_{i=1}^{r} a_i \cdot (y_i' - y_i) \;+\; \sum_{\ell \in \mathcal{L}} b_\ell \cdot (at'\_\ell - at\_\ell) \;=\; 0$$

resulting in the set of equations

$$\boxed{\sum_{i=1}^{r} a_i \cdot \Delta(y_i, \tau) \;+\; \sum_{\ell \in \mathcal{L}} b_\ell \cdot \Delta(at\_\ell, \tau) \;=\; 0}$$

for every transition $\tau \in \mathcal{T}$

`Example:` Program DOUBLE

local y: **integer where** $y = 0$

$$\begin{bmatrix} \ell_0\colon\ y := y+1 \\ \ell_1\colon \end{bmatrix} \;\|\; \begin{bmatrix} m_0\colon\ y := y+1 \\ m_1\colon \end{bmatrix}$$

linear invariant:

$$\varphi\colon\ a \cdot y + b_{\ell_0} \cdot at\_\ell_0 + b_{\ell_1} \cdot at\_\ell_1 + b_{m_0} \cdot at\_m_0 + b_{m_1} \cdot at\_m_1 = K$$

(I) $a \cdot 0 + b_{\ell_0} + b_{m_0} = K$
(initial value of y is 0)

(T) $a \cdot 1 - b_{\ell_0} + b_{\ell_1} = 0$ (for ℓ_0)
$a \cdot 1 - b_{m_0} + b_{m_1} = 0$ (for m_0)

`Example:` Program DOUBLE (Con'd)

Possible solutions (basis for all solutions)

	a	b_{ℓ_0}	b_{ℓ_1}	b_{m_0}	b_{m_1}	K
S_1	0	1	1	0	0	1
S_2	0	0	0	1	1	1
S_3	1	1	0	1	0	2

Corresponding invariants

φ_1: $at\_\ell_0 + at\_\ell_1 = 1$ (control invariant)

φ_2: $at\_m_0 + at\_m_1 = 1$ (control invariant)

$\boxed{\varphi_3\colon\ y + at\_\ell_0 + at\_m_0 = 2}$

Linear Invariants for Cyclic Programs

Program $\ell_0^1\colon\ S_1 \;\|\; \ldots \;\|\; \ell_0^j\colon\ S_j \;\|\; \ldots \;\|\; \ell_0^m\colon\ S_m$
where S_j is of the form

ℓ_0^j: **loop forever do** $\underbrace{\ell_1^j, \ell_2^j, \ldots, \ell_k^j}_{\text{cycle } C}$

Define

$$\Delta(y, C) = \Delta(y, \tau_1) + \cdots + \Delta(y, \tau_k)$$

For these programs construction of the linear invariants can be done in three phases:

1. Compute a_i's
2. Compute b_ℓ's
3. Compute K

Phase 1: Bodies

For cycle $\underbrace{\ell_1, \ell_2, \ldots, \ell_k}_{C}$

$$\sum_{i=1}^{r} a_i \cdot \Delta(y_i, \tau_{\ell_1}) - b_{\ell_1} + b_{\ell_2} = 0$$

$$\sum_{i=1}^{r} a_i \cdot \Delta(y_i, \tau_{\ell_2}) - b_{\ell_2} + b_{\ell_3} = 0$$

$$\vdots$$

$$\sum_{i=1}^{r} a_i \cdot \Delta(y_i, \tau_{\ell_k}) + b_{\ell_1} - b_{\ell_k} = 0$$

$$\sum_{i=1}^{r} a_i \cdot (\Delta(y_i, \tau_{\ell_1}) + \ \ldots \ + \Delta(y_i, \tau_{\ell_k})) = 0$$

Thus,

$$\boxed{\sum_{i=1}^{r} a_i \cdot \Delta(y_i, C) = 0}$$

Phase 2: Compensation Expressions

$$\boxed{b_{\ell_0} = 0}$$

For $\tau: \ell_j \rightarrow \ell_k$ where $j < k$

$$\sum_{i=1}^{r} a_i \cdot \Delta(y_i, \tau) \; - \; b_{\ell_j} \; + \; b_{\ell_k} \; = \; 0$$

Assume that for all $j < k$, b_{ℓ_j} is known.
Compute b_{ℓ_k} from

$$\boxed{b_{\ell_k} \; = \; b_{\ell_j} \; - \; \sum_{i=1}^{r} a_i \cdot \Delta(y_i, \tau)}$$

(independently for each cycle)

Phase 3: Right constants

$$\boxed{K \; = \; \sum_{i=1}^{r} a_i \cdot y_i^0}$$

Note: This set of equations has the same solutions as the equations (T) + (I) except for solutions of the form

$$at\_\ell_1 + \; \cdots \; + \; at\_\ell_k = 1$$

which are produced by (T) + (I), but not by this set.

`Example:` Program PROD-CON-SV (Fig 2.23)
Producer-Consumer with
shared variables

- semaphores r, ne, nf:

  ne – counts # of empty slots in list b
  initially $ne = N$

  nf – counts # of full slots in b
  initially $nf = 0$

  r – ensures that the shared variable b is handled exclusively by *Prod* or *Cons*

- linear variables: $r, ne, nf, |b|$

Program PROD-CONS-SV (Fig. 2.23)

local $r,\ ne,\ nf$: **integer where** $r = 1,\ ne = N,\ nf = 0$
b : **list of integer where** $b = \Lambda$

```
Prod :: [ local x: integer
          ℓ0: loop forever do
                [ ℓ1: produce x
                  ℓ2: request ne
                  ℓ3: request r
                  ℓ4: b := b • x
                  ℓ5: release r
                  ℓ6: release nf ] ]
||
Cons :: [ local y: integer
          m0: loop forever do
                [ m1: request nf
                  m2: request r
                  m3: (y, b) := (hd(b), tl(b))
                  m4: release r
                  m5: release ne
                  m6: consume y ] ]
```

Properties we want to prove:

$\square \underbrace{\neg(at\_\ell_4 \ \wedge \ at\_m_3)}_{\psi_1}$

$\square \underbrace{at\_\ell_4 \ \rightarrow \ |b| < N}_{\psi_2}$

$\square \underbrace{at\_m_3 \ \rightarrow \ |b| > 0}_{\psi_3}$

Bottom-up invariants:

$\underbrace{r \geq 0}_{\varphi_0} \wedge \underbrace{ne \geq 0}_{\varphi_1} \wedge \underbrace{nf \geq 0}_{\varphi_2} \wedge \underbrace{|b| \geq 0}_{\varphi_3}$

Bodies:

Increments along each cycle:

	Prod	Cons
r	0	0
ne	-1	1
nf	1	-1
$\|b\|$	1	-1

For each cycle: $\sum_{i=1}^{r} a_i \cdot \Delta(y_i, C) \ = \ 0$

Therefore

Prod: $-a_e + a_f + a_b = 0$

Cons: $a_e - a_f - a_b = 0$

Solutions			Bodies
1.	$a_r = 1,$	$a_e = a_f = a_b = 0$	B_1: r
2.	$a_e = a_f = 1,$	$a_r = a_b = 0$	B_2: $ne + nf$
3.	$a_e = a_b = 1,$	$a_r = a_f = 0$	B_3: $ne + \|b\|$

compensation expressions

coefficients of $b_{\ell_1}, \ldots, b_{m_6}$
corresponding to bodies
B_1: r
B_2: $ne + nf$
B_3: $ne + |b|$

– Prod –				– Cons –			
	B_1	B_2	B_3		B_1	B_2	B_3
b_{ℓ_1}	0	0	0	b_{m_1}	0	0	0
b_{ℓ_2}	0	0	0	b_{m_2}	0	1	0
b_{ℓ_3}	0	1	1	b_{m_3}	1	1	0
b_{ℓ_4}	1	1	1	b_{m_4}	1	1	1
b_{ℓ_5}	1	1	0	b_{m_5}	0	1	1
b_{ℓ_6}	0	1	0	b_{m_6}	0	0	0

Right constants

$b_{\ell_0} = b_{m_0} = 0$
Initial values

$$r = 1, \ ne = N, \ nf = 0, \ |b| = 0$$

$$K_1 \ = \ 1 \cdot \underbrace{1}_{r} \ = \ 1$$

$$K_2 \ = \ 1 \cdot \underbrace{N}_{ne} + 1 \cdot \underbrace{0}_{nf} \ = \ N$$

$$K_3 \ = \ 1 \cdot \underbrace{N}_{ne} + 1 \cdot \underbrace{0}_{|b|} \ = \ N$$

The resulting invariants

α_1: $r + at\_\ell_{4,5} + at\_m_{3,4} \ = \ 1$

α_2: $ne + nf + at\_\ell_{3..6} + at\_m_{2..5} \ = \ N$

α_3: $ne + |b| + at\_\ell_{3,4} + at\_m_{4,5} \ = \ N$

No need to check invariance!

These invariants imply the properties we wanted to prove:

$$\psi_1: \quad \underbrace{r + at\_\ell_{4,5} + at\_m_{3,4} = 1}_{\alpha_1} \wedge \underbrace{r \geq 0}_{\varphi_0} \quad \rightarrow \underbrace{\neg(at\_\ell_4 \wedge at\_m_4)}_{\psi_1}$$

$$\psi_2: \quad \underbrace{ne + |b| + at\_\ell_{3,4} + at\_m_{4,5} = N}_{\alpha_3} \wedge \underbrace{ne \geq 0}_{\varphi_1} \quad \rightarrow \underbrace{at\_\ell_4 \rightarrow |b| < N}_{\psi_2}$$

Since $at\_\ell_4 \rightarrow at\_\ell_{3,4} = 1$
and $ne \geq 0$, $at\_\ell_{3,4} = 1$, $at\_m_{4,5} \geq 0$ implies $|b| < N$

$$\psi_3: \quad \underbrace{ne + nf + at\_\ell_{3..6} + at\_m_{2..5} = N}_{\alpha_2} \wedge \underbrace{ne + |b| + at\_\ell_{3,4} + at\_m_{4,5} = N}_{\alpha_3} \wedge \underbrace{nf \geq 0}_{\varphi_2} \quad \rightarrow \underbrace{at\_m_3 \rightarrow |b| > 0}_{\psi_3}$$

Suppose at_m_3:

φ_2: $ne + nf + at\_\ell_{3..6} + 1 = N$

φ_3: $ne + |b| + at\_\ell_{3,4} + 0 = N$

Since $\varphi_2 - \varphi_3$ yields
$nf - |b| + at\_\ell_{3..6} - at\_\ell_{3,4} + 1 = 0$

Thus
$$|b| = \underbrace{nf}_{\geq 0} + \underbrace{(at\_\ell_{3..6} - at\_\ell_{3,4})}_{\geq 0} + 1 > 0$$