Verification – Lecture 5
Parameterized Programs, Linear Invariants

Bernd Finkbeiner – Sven Schewe
Rayna Dimitrova – Lars Kuhtz – Anne Proetzsch

Wintersemester 2007/2008

Proving Invariance Properties

Review

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

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

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

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

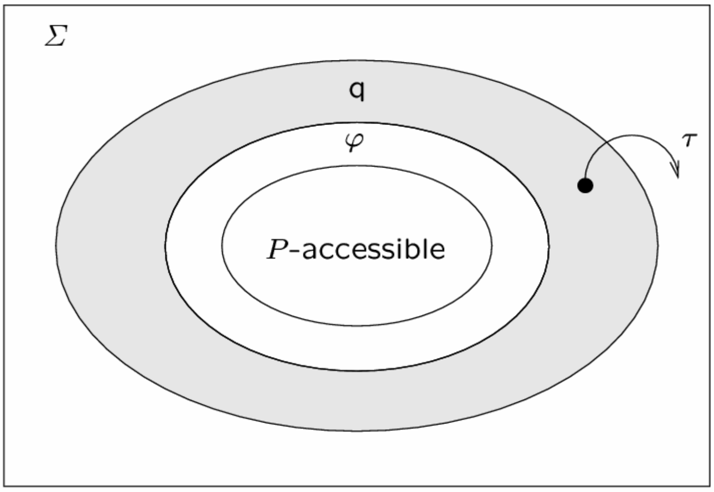
I3. $P \Vdash \{(\bigwedge_{i=1}^{k} \chi_i) \land \varphi\}\ \mathcal{T}\ \{\varphi\}$

$P \vDash \Box\, q$

INC-INV

Strategy 1: Strengthening

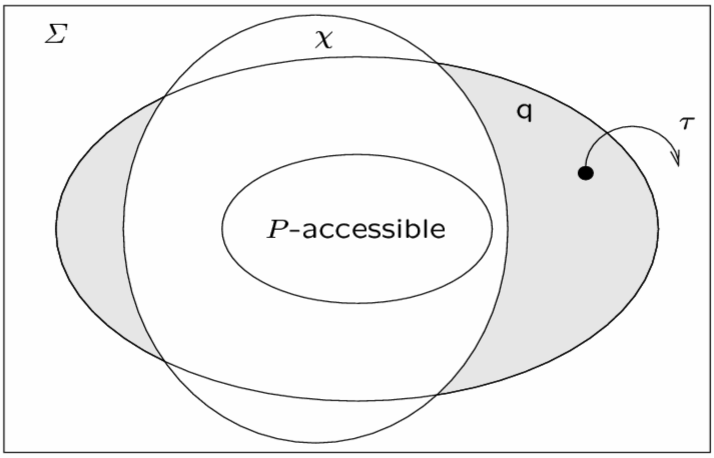
Review



Find a stronger assertion φ that is inductive and implies the assertion q we want to prove.

Strategy 2: Incremental Proofs

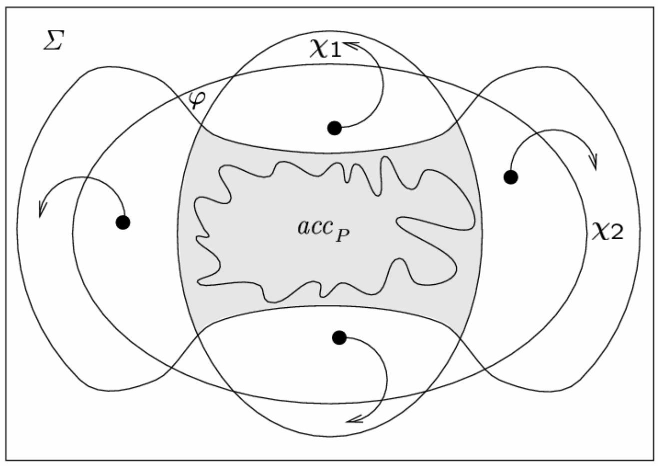
Review



Use previously proven invariances χ to exclude parts of the state space from consideration.

Discussion

Review



Although the assertion acc_P is inductive and strengthens any P-invariant, it is not very useful in practice.

Finding Inductive Invariants

Review

Construction of inductive assertions by

1. Bottom-up methods:
   - Based on program text only
   - Algorithmic
   - Guaranteed to produce an inductive invariant
2. Top-down methods:
   - Guided by the property we want to prove
   - Heuristic
   - Not guaranteed to produce an inductive invariant

Top-Down Approach

Review

previously proven $\square \chi$

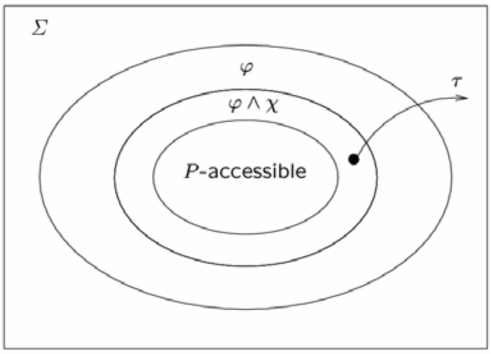
want to prove $\square \varphi$

but

$$\{\chi \wedge \varphi\}\tau\{\varphi\}$$

not state-valid

for some $\tau \in \mathcal{T}$.

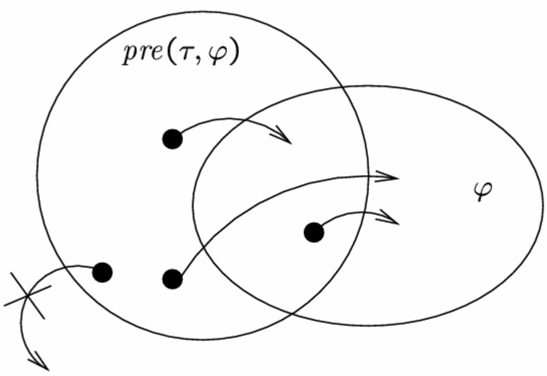


Solution: Take the largest set of states that will result in a φ-state when τ is taken.

Precondition

Review

$$pre(\tau, \varphi) \; : \; \forall V' . \rho_\tau \rightarrow \varphi'$$



a state s satisfies $pre(\tau, \varphi)$

iff

all τ-successors of s satisfy φ.

Heuristic

Review

If the verification condition

$$\{\chi \wedge \varphi\}\tau\{\varphi\}$$

is not state-valid

- Strengthening approach:
  strengthen φ by adding the conjunct $pre(\tau, \varphi)$

- Incremental approach:
  prove $\Box\, pre(\tau, \varphi)$ and add $pre(\tau, \varphi)$ to χ.

Example

Review

local y_1, y_2: boolean where $y_1 = \text{F}, y_2 = \text{F}$
s : integer where $s = 1$

P_1 ::
ℓ_0 : loop forever do
- ℓ_1 : noncritical
- ℓ_2 : $(y_1,\ s) := (\text{T},\ 1)$
- ℓ_3 : await $(\neg y_2) \vee (s = 2)$
- ℓ_4 : critical
- ℓ_5 : $y_1 := \text{F}$

||

P_2 ::
m_0 : loop forever do
- m_1 : noncritical
- m_2 : $(y_2,\ s) := (\text{T},\ 2)$
- m_3 : await $(\neg y_1) \vee (s = 1)$
- m_4 : critical
- m_5 : $y_2 := \text{F}$

Goal:

Mutual Exclusion for Peterson's algorithm:

$$\Box \underbrace{\neg(at\_\ell_4 \ \wedge \ at\_m_4)}_{\psi}$$

Bottom-up invariants:

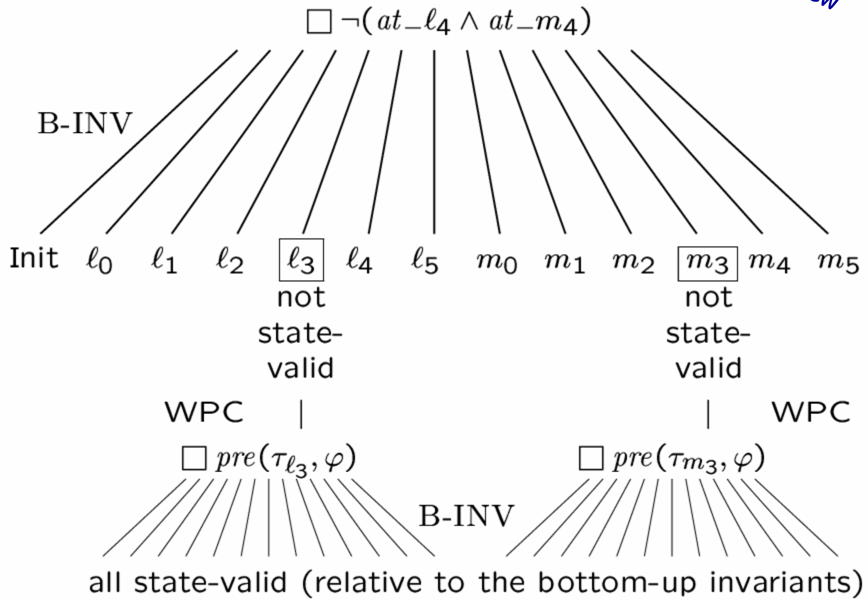
φ_0: $s = 1 \ \vee \ s = 2$

φ_1: $y_1 \ \leftrightarrow \ at\_\ell_{3..5}$

φ_2: $y_2 \ \leftrightarrow \ at\_m_{3..5}$

Example (cont'd)

Review



Goals for Today

1. Generalization to parameterized systems
2. Bottom-up generation of linear invariants

Introduction: Parameterized Programs

$$S:: \left[\begin{array}{l} \ell_0\colon \textbf{loop forever do} \\ \quad \left[\begin{array}{l} \ell_1\colon \textbf{noncritical} \\ \ell_2\colon \textbf{request } y \\ \ell_3\colon \textbf{critical} \\ \ell_4\colon \textbf{release } y \end{array}\right] \end{array}\right]$$

$P^3::$ [**local** y : **integer where** $y = 1$; $[S||S||S]$]
(with some renaming of labels of the S's.)

$P^4::$ [**local** y : **integer where** $y = 1$; $[S||S||S||S]$]

⋮

$P^n::$?

Introduction: Parameterized Specifications

Mutual exclusion:

P^3: $\Box(\neg(at\_\ell_3 \wedge at\_m_3) \wedge \neg(at\_\ell_3 \wedge at\_k_3) \wedge \neg(at\_m_3 \wedge at\_k_3))$

P^4: $\Box(\neg(\ldots) \wedge \ldots \wedge \neg(\ldots))$

P^n: ?

We want to deal with these programs, i.e., programs with an arbitrary number of identical components, in a more uniform way.

Syntax

cooperation: $\overset{M}{\underset{j=1}{\|}} S[j] \quad : \quad [\ S[1] \| \ldots \| S[M]\ ]$

Selection: $\overset{M}{\underset{j=1}{\textbf{OR}}} S[j] \quad : \quad [\ S[1]\ \textbf{or}\ \ldots\ \textbf{or}\ S[M]\ ]$

$S[j]$ is a parameterized statement.

Parameterized Statements

- explicit variable in expression

  $\ldots := j + \ldots$

- explicit subscript in array x

  $\ldots := x[j] + \ldots$ or $x[j] := \ldots$

- implicit subscript of all local variables in $S[j]$

  z stands for $z[j]$

- implicit subscript of all labels in $S[j]$

  ℓ_3 stands for $\ell_3[j]$

Example: Program PAR-SUM

$$
\begin{array}{ll}
\textbf{in} & M\text{: } \textbf{integer where } M \geq 1 \\
 & x\ \text{: } \textbf{array } [1..M] \textbf{ of integer} \\
\textbf{out} & z\ \text{: } \textbf{integer where } z = 0
\end{array}
$$

$$
\mathop{\Big\|}_{j=1}^{M} P[j] :: \left[\begin{array}{l}
\textbf{local } y\text{: } \textbf{integer} \\
\ell_0\text{: } y := x[j] \\
\ell_1\text{: } z := z + y \cdot y \\
\ell_2\text{:}
\end{array}\right]
$$

Parallel sum of squares:

$$z \;=\; x[1]^2 + x[2]^2 + \ldots + x[M]^2$$

Parameterized Transition Systems

The number M of processes is not fixed, so there is an unbounded # of transitions. To finitely represent these, we use parameterization of transition relations.

Example: PAR-SUM

The unbounded # of transitions associated with ℓ_0 are represented by a single transition relation using parameter j:

$$\rho_{\ell_0}[j]: \quad move(\ell_0[j], \ell_1[j]) \;\wedge\; y'[j] = x[j] \qquad j = 1 \ldots M$$

Arrays

Arrays (explicit or implicit) are treated as variables that range over functions:
$[1 \ldots M] \mapsto$ integers

Representation of array operations in transition relations:

- Retrieval: $y[k]$
  to retrieve the value of the kth element of array y
- Modification: $update(y, k, e)$
  the resulting array agrees with y on all i, $i \neq k$, and $y[k] = e$

Arrays (cont'd)

Properties of *update*

$$update(y, k, e)[k] = e$$
$$update(y, k, e)[j] = y[j] \text{ for } j \neq k$$

Example: PAR-SUM

The proper representation of the transition relation for $\ell_0[j]$ is

$$\rho_0[j]: \quad move(\ell_0[j],\ \ell_1[j]) \wedge \\ y' = update(y,\ j,\ x[j]) \wedge \\ pres(\{x, z\})$$

Notation in Specifications

- $L_i = \{j \mid \ell_i[j] \in \pi\} \quad \subseteq \{1, \ldots, M\}$

  The set of indices of processes that currently reside at ℓ_i

- $N_i = |L_i|$

  The number of processes currently residing at ℓ_i

Abbreviations

$$
\begin{aligned}
L_{i_1,i_2,\ldots,i_k} &= L_{i_1} \cup L_{i_2} \cup \ldots \cup L_{i_k} \\
L_{i..j} &= L_i \cup L_{i+1} \cup \ldots \cup L_j \\
N_{i_1,i_2,\ldots,i_k} &= |L_{i_1,i_2,\ldots,i_k}| \\
N_{i..j} &= |L_{i..j}|
\end{aligned}
$$

Example: MPX-SEM

in M: **integer where** $M \geq 2$
local y : **array** $[1..M]$ **of integer**
where $y[1] = 1,\ y[j] = 0$ for $2 \leq j \leq M$

$$\overset{M}{\underset{j=1}{||}} P[j] :: \left[\begin{array}{l} \ell_0\text{: loop forever do} \\ \quad \left[\begin{array}{l} \ell_1\text{: noncritical} \\ \ell_2\text{: request } y[j] \\ \ell_3\text{: critical} \\ \ell_4\text{: release } y[j \oplus_M 1] \end{array} \right] \end{array} \right]$$

Multiple mutual exclusion by semaphors

$$j \oplus_M 1 = (j \bmod M) + 1 = \begin{cases} j+1 & \text{if } j < M \\ 1 & \text{if } j = M \end{cases}$$

Example: Specification

in M: **integer where** $M \geq 2$
local y : **array** $[1..M]$ **of integer**
where $y[1] = 1,\ y[j] = 0$ for $2 \leq j \leq M$

$$\overset{M}{\underset{j=1}{||}} P[j] :: \left[\begin{array}{l} \ell_0\text{: loop forever do} \\ \quad \left[\begin{array}{l} \ell_1\text{: noncritical} \\ \ell_2\text{: request } y[j] \\ \ell_3\text{: critical} \\ \ell_4\text{: release } y[j \oplus_M 1] \end{array} \right] \end{array} \right]$$

mutual exclusion:

$$\Box \underbrace{\forall i,j \in [1..M] \,.\, i \neq j \,.\, \neg(at\_\ell_3[i] \ \wedge \ at\_\ell_3[j])}_{\psi}$$

abbreviated as

$$\boxed{\Box(N_3 \leq 1)}$$

Example: Verification

in M: **integer where** $M \geq 2$
local y : **array** $[1..M]$ **of integer**
where $y[1] = 1,\ y[j] = 0$ for $2 \leq j \leq M$

$$\mathop{\Big\|}_{j=1}^{M} P[j] :: \left[\begin{array}{l} \ell_0\text{: loop forever do} \\ \quad \left[\begin{array}{l} \ell_1\text{: noncritical} \\ \ell_2\text{: request } y[j] \\ \ell_3\text{: critical} \\ \ell_4\text{: release } y[j \oplus_M 1] \end{array}\right] \end{array}\right]$$

$$\Box(\underbrace{N_3 \leq 1}_{\varphi})$$

The assertion φ is not inductive, therefore we prove the invariance of

$$\varphi_1: \quad \forall j \,.\, y[j] \geq 0$$

$$\varphi_2: \quad \Big(N_{3,4} + \sum_{j=1}^{M} y[j]\Big) = 1$$

Example (cont'd)

$$\varphi_1: \quad \forall j \,.\, y[j] \geq 0$$

$$\varphi_2: \quad \Big(N_{3,4} + \sum_{j=1}^{M} y[j]\Big) = 1$$

$$\Box(\underbrace{N_3 \leq 1}_{\varphi})$$

Then φ can be deduced by monotonicity:

$$\varphi_1 \ \wedge \ \varphi_2 \ \rightarrow \ \underbrace{N_3 \leq 1}_{\varphi}$$

since

$$N_3 \ \leq \ N_{3,4} \ \underset{\varphi_2}{=} \ 1 - \sum_{j=1}^{M} y[j] \ \underset{\varphi_1}{\leq} \ 1$$

Example (cont'd)

- Proof of $\square(\underbrace{\forall j\,.\,y[j] \geq 0}_{\varphi_1})$

B1:

$$\underbrace{\ldots \wedge\ y[1]=1\ \wedge\ (\forall j\,.\,2 \leq j \leq M\,.\,y[i]=0)}_{\Theta}$$
$$\rightarrow\ \underbrace{\forall j\,.\,y[i] \geq 0}_{\varphi_1}$$

B2:
The only transitions that interfere with φ_1 are τ_{ℓ_2} and τ_{ℓ_4}.

Example (cont'd)

$$\rho_{\ell_2}:\ move(\ell_2[i],\ell_3[i])\ \wedge\ y[i]>0\ \wedge$$
$$y' = update(y,i,y[i]\text{-}1)$$
$$\rho_{\ell_4}:\ move(\ell_4[i],\ell_0[i])\ \wedge$$
$$y' = update(y,i \oplus_M 1, y[i \oplus_M 1]{+}1)$$

$\rho_{\ell_2}[i]$ implies

$$y[i]>0\ \wedge y'[i]=y[i]-1\ \wedge\ \forall j\,.\,j \neq i\,.\,y'[j]=y[j]$$

$\rho_{\ell_4}[i]$ implies

$$y'[i \oplus_M 1] = y[i \oplus_M 1]+1\ \wedge$$
$$\forall j\,.\,j \neq i \oplus_M 1\,.\,y'[j]=y[j]$$

therefore $\underbrace{\forall j\,.\,y[i] \geq 0}_{\varphi_1} \wedge \left\{ \begin{array}{c} \rho_{\ell_2}[i] \\ \rho_{\ell_4}[i] \end{array} \right\} \rightarrow \underbrace{\forall j\,.\,y'[j] \geq 0}_{\varphi_1'}$

Example (cont'd)

- Proof of $\underbrace{\Box\left(N_{3,4} + \left(\sum_{j=1}^{M} y[j]\right) = 1\right)}_{\varphi_2}$

B1:

$$\underbrace{\begin{pmatrix} \pi = \{\ell_0[1], \ldots, \ell_0[M]\} \wedge \\ y[1] = 1 \;\wedge\; (\forall j \,.\, 2 \le j \le M \,.\, y[i] = 0) \end{pmatrix}}_{\Theta}$$

$$\rightarrow \underbrace{N_{3,4} + \left(\sum_{j=1}^{M} y[j]\right) = 1}_{\varphi_2}$$

Example (cont'd)

ρ_{ℓ_2}: $move(\ell_2[i], \ell_3[i]) \;\wedge\; y[i] > 0 \wedge$
$y' = update(y, i, y[i]\text{-}1)$

ρ_{ℓ_4}: $move(\ell_4[i], \ell_0[i]) \wedge$
$y' = update(y, i \oplus_M 1, y[i \oplus_M 1]{+}1)$

B2: Verification conditions:

$\rho_{\ell_2}[i]$ implies:

$$N'_{3,4} = N_{3,4} + 1 \;\wedge\; \left(\sum_{j=1}^{M} y'[i]\right) = \left(\sum_{j=1}^{M} y[i]\right) - 1$$

Example (cont'd)

ρ_{ℓ_4}: $move(\ell_4[i], \ell_0[i]) \wedge$
$y' = update(y, i \oplus_M 1, y[i \oplus_M 1]+1)$

$\rho_{\ell_4}[i]$ implies:

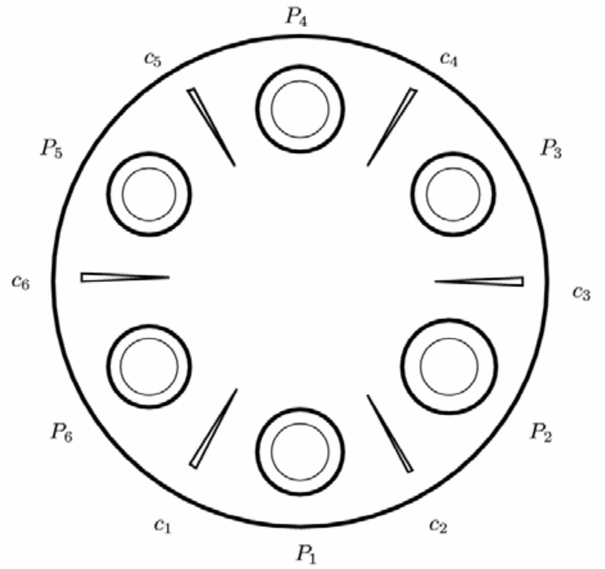
$$N'_{3,4} = N_{3,4} - 1 \ \wedge \ \left(\sum_{j=1}^{M} y'[i]\right) = \left(\sum_{j=1}^{M} y[i]\right) + 1$$

Therefore

$$\underbrace{N_{3,4} + \left(\sum_{j=1}^{M} y[i]\right) = 1}_{\varphi_2} \ \wedge \ \left\{ \begin{array}{c} \rho_{\ell_2}[i] \\ \rho_{\ell_4}[i] \end{array} \right\}$$

$$\rightarrow \underbrace{N'_{3,4} + \left(\sum_{j=1}^{M} y[i]\right) = 1}_{\varphi'_2}$$

The Dining Philosophers



DINE – A Simple Solution

in M: integer where $M \geq 2$
local c : array $[1..M]$ of integer where $c = 1$

$$\mathop{||}_{j=1}^{M} P[j] :: \left[\begin{array}{l} \ell_0\text{: loop forever do} \\ \quad \left[\begin{array}{l} \ell_1\text{: noncritical} \\ \ell_2\text{: request } c[j] \\ \ell_3\text{: request } c[j \oplus_M 1] \\ \ell_4\text{: critical} \\ \ell_5\text{: release } c[j] \\ \ell_6\text{: release } c[j \oplus_M 1] \end{array}\right] \end{array}\right]$$

„Chopstick Exclusion"

$$\Box \underbrace{\forall j \in [1..M] \,.\, \neg(at\_\ell_4[j] \;\wedge\; at\_\ell_4[j \oplus_M 1])}_{\psi}$$

Mutual exclusion between every two adjacent philosophers

- φ_0 and φ_1 are inductive

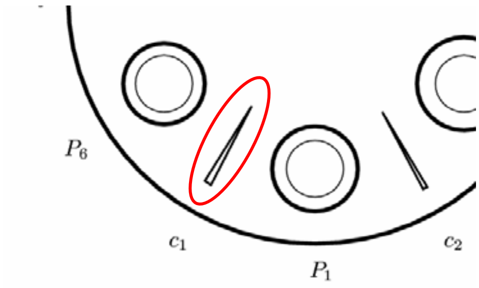
  φ_0: $\forall j \in [1..M] \,.\, c[j] \geq 0$

  φ_1: $\forall j \in [1..M] \,.\, at\_\ell_{4..6}[j] \;+\; at\_\ell_{3..5}[j \oplus_M 1] \;+\; c[j \oplus_M 1] \;=\; 1$

- Then,

$$at\_\ell_4[j] + at\_\ell_4[j \oplus_M 1] \underset{\varphi_1}{\leq} 1 - c[j \oplus_M 1] \underset{\varphi_0}{\leq} 1$$

Deadlock!

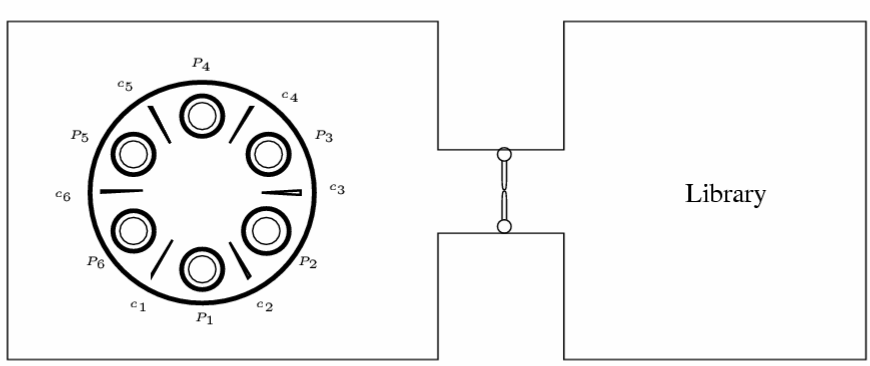


$P[1]$ $\quad \ell_2$: **request** $c[1]$; $\quad \ell_3$: **request** $c[2]$
$\uparrow$

.
.
.

$P[M]$ $\quad \ell_2$: **request** $c[M]$; $\quad \ell_3$: **request** $c[1]$
$\uparrow$

Two-Room Philosopher's World



No more than $M-1$ philosophers are admitted to the dining hall at the same time.

Program DINE-EXCL

$$
\begin{array}{ll}
\textbf{in} & M\colon \textbf{integer where } M \geq 2 \\
\textbf{local } c & : \textbf{array } [1..M] \textbf{ integer where } c = 1 \\
\quad\quad r & : \textbf{integer where } r = M - 1
\end{array}
$$

$$
\mathop{||}_{j=1}^{M} P[j] :: \left[\begin{array}{l}
\ell_0\colon \textbf{loop forever do} \\
\quad \left[\begin{array}{l}
\ell_1\colon \textbf{noncritical} \\
\ell_2\colon \textbf{request } r \\
\ell_3\colon \textbf{request } c[j] \\
\ell_4\colon \textbf{request } c[j \oplus_M 1] \\
\ell_5\colon \textbf{critical} \\
\ell_6\colon \textbf{release } c[j] \\
\ell_7\colon \textbf{release } c[j \oplus_M 1] \\
\ell_8\colon \textbf{release } r
\end{array}\right]
\end{array}\right]
$$

Finding Inductive Invariants

Review

Construction of inductive assertions by

1. Bottom-up methods:
   - Based on program text only
   - Algorithmic
   - Guaranteed to produce an inductive invariant
2. Top-down methods:
   - Guided by the property we want to prove
   - Heuristic
   - Not guaranteed to produce an inductive invariant

Control Invariants

Review

- CONFLICT:
  for labels ℓ_i, ℓ_j that are in conflict
  (i.e., not $\sim_L$, not parallel):

  $\Box \neg(at\_\ell_i \wedge at\_\ell_j)$

- SOMEWHERE:
  for the set of labels $\mathcal{L}_i$ in a
  top-level process:

  $\Box \bigvee_{\ell \in \mathcal{L}_i} at\_\ell$

Control Invariants

Review

- EQUAL:
  for labels l, m, s.t. $l \sim_L m$:

  $\Box(at\_\ell \leftrightarrow at\_m)$

- PARALLEL:
  for substatement $[S_1 || S_2]$:

  $\Box(in\_S_1 \leftrightarrow in\_S_2)$

  i.e, if control is in S_1 it must also be in S_2
  and vice versa.

Transition-Validated Assertions

Review

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

if no statement parallel to ℓ_1 can modify variables in c

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

if no statement parallel to ℓ_1 can modify y or variables occurring in e
and if y does not occur in e.

Single-Variable Assertions

Review

$$y = 0$$

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

$\boxed{y \geq 0}$

$$s = 1$$

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

$\boxed{s = 1 \;\vee\; s = 2}$

where no other statement modifies s

Linear Variables

Definition: integer variable y is linear in P if

$$y' = y + c \qquad \text{for every } \rho_\tau$$

where c is some integer constant

Example: semaphore variables are linear

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

Linear Invariants

A linear invariant is of the form

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

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

$$\textbf{local } y\text{: } \textbf{integer where } y = 0$$

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

linear variable: y

linear invariant:

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

Assumptions

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

- 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 $\rho_\tau \rightarrow at'\_\ell = at\_\ell + \Delta(at\_\ell,\tau)$

Automatic Invariant Construction

Construct

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

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

Equations

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

Θ implies $y_i = y_i^0 \ (i = 1 \ldots r)$
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 (cont'd)

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

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

Equations (cont'd)

$$\underbrace{\left(\sum_{i=1}^{r} a_i \cdot y_i \;+\; \sum_{\ell\in\mathcal{L}} b_\ell \cdot at\_\ell = K\right)}_{\varphi} \wedge \rho_\mathcal{T}$$
$$\rightarrow \underbrace{\left(\sum_{i=1}^{r} a_i \cdot y_i' \;+\; \sum_{\ell\in\mathcal{L}} b_\ell \cdot at'\_\ell = K\right)}_{\varphi'}$$

or $\quad \rho_\mathcal{T} \;\rightarrow\; \sum_{i=1}^{r} a_i \cdot (y_i' - y_i) \;+\; \sum_{\ell\in\mathcal{L}} b_\ell \cdot (at'\_\ell - at\_\ell) \;=\; 0$

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 $y = 0$

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

linear invariant:

φ: $\quad a \cdot y \;+\; b_{\ell_0} \cdot at\_\ell_0 \;+\; b_{\ell_1} \cdot at\_\ell_1 \;+$
$\quad b_{m_0} \cdot at\_m_0 \;+\; b_{m_1} \cdot at\_m_1 \;=\; K$

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

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

Example (cont'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)

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

Linear Invariants for Cyclic Programs

Program ℓ_0^1: S_1 $\|$ $\ldots \|$ ℓ_0^j: S_j $\|$ $\ldots$ $\|$ ℓ_0^m: S_m

where S_j is of the form

$$\ell_0^j\text{: } \textbf{loop forever do } \underbrace{\ell_1^j, \ell_2^j, \ldots, \ell_k^j}_{\text{cycle } C}$$

Define

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

Example: PRODUCER-CONSUMER

$$\begin{array}{l} \textbf{local } r,\ ne,\ nf: \textbf{integer where } r = 1,\ ne = N,\ nf = 0 \\ \quad b \quad : \textbf{list of integer where } b = \Lambda \end{array}$$

$$\left[\begin{array}{l} \textbf{local } x\text{: } \textbf{integer} \\ \ell_0\text{: } \textbf{loop forever do} \\ \quad \left[\begin{array}{l} \ell_1\text{: } \textbf{produce } x \\ \ell_2\text{: } \textbf{request } ne \\ \ell_3\text{: } \textbf{request } r \\ \ell_4\text{: } b := b \bullet x \\ \ell_5\text{: } \textbf{release } r \\ \ell_6\text{: } \textbf{release } nf \end{array}\right] \end{array}\right] \quad \| \quad \left[\begin{array}{l} \textbf{local } y\text{: } \textbf{integer} \\ m_0\text{: } \textbf{loop forever do} \\ \quad \left[\begin{array}{l} m_1\text{: } \textbf{request } nf \\ m_2\text{: } \textbf{request } r \\ m_3\text{: } (y,\ b) := (hd(b),\ tl(b)) \\ m_4\text{: } \textbf{release } r \\ m_5\text{: } \textbf{release } ne \\ m_6\text{: } \textbf{consume } y \end{array}\right] \end{array}\right]$$

(Prod) (Cons)

Invariant Construction

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

3 Phases:

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

Phase 1: Bodies

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

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

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

$$\vdots$$

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

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

Phase 2: Compensation Expressions

$$b_{\ell_0} = 0$$

For $\tau : \ell_j \to \ell_k$ 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

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

(independently for each cycle)

Phase 3: Right Constants

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

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

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

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

Example: PRODUCER-CONSUMER

```
local r, ne, nf : integer where r = 1, ne = N, nf = 0
      b         : list of integer where b = Λ

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

Increments along each cycle:

	Prod	Cons
r	0	0
ne	-1	1
nf	1	-1
$\lvert b\rvert$	1	-1

Example: PRODUCER-CONSUMER

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

$$\left[\begin{array}{l} \textbf{local } x\text{: } \textbf{integer} \\ \ell_0\text{: } \textbf{loop forever do} \\ \quad \left[\begin{array}{l} \ell_1\text{: } \textbf{produce } x \\ \ell_2\text{: } \textbf{request } ne \\ \ell_3\text{: } \textbf{request } r \\ \ell_4\text{: } b := b \bullet x \\ \ell_5\text{: } \textbf{release } r \\ \ell_6\text{: } \textbf{release } nf \end{array}\right] \end{array}\right] \quad \| \quad \left[\begin{array}{l} \textbf{local } y\text{: } \textbf{integer} \\ m_0\text{: } \textbf{loop forever do} \\ \quad \left[\begin{array}{l} m_1\text{: } \textbf{request } nf \\ m_2\text{: } \textbf{request } r \\ m_3\text{: } (y,\ b) := (hd(b),\ tl(b)) \\ m_4\text{: } \textbf{release } r \\ m_5\text{: } \textbf{release } ne \\ m_6\text{: } \textbf{consume } y \end{array}\right] \end{array}\right]$$

We look for linear invariants with the body

$$a_r \cdot r \ + \ a_e \cdot ne \ + \ a_f \cdot nf \ + \ a_b \cdot |b|$$

Example (cont'd)

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: $a_e - a_f - a_b = 0$

<u>Solutions</u>			<u>Bodies</u>
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: $ne + \|b\|$

Example (cont'd)

```
local r, ne, nf: integer where r = 1, ne = N, nf = 0
      b        : list of integer where b = Λ

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

compensation expressions

coefficients of $b_{\ell_1}, \ldots, b_{m_6}$
(corresponding to bodies B_1, B_2, B_3)

– 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_2}	0	1	1
b_{ℓ_6}	0	1	0	b_{m_6}	0	0	0

Bodies

B_1: r

B_2: $ne + nf$

B_3: $ne + |b|$

Example (cont'd)

Right constants

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

φ_1: $r + at\_\ell_{4,5} + at\_m_{3,4} = 1$

φ_2: $ne + nf + at\_\ell_{3..6} + at\_m_{2..5} = N$

φ_3: $ne + |b| + at\_\ell_{3,4} + at\_m_{4,5} = N$