Finding Inductive Assertions
Top-Down Approach

Assertion propagation

we have previously proven $\Box \chi$
and we want to prove $\Box \varphi$
but

$\{\chi \land \varphi\} \tau \{\varphi\}$

is not state-valid for some $\tau \in \mathcal{T}$.

What is the problem?
(assuming that $\varphi$ is indeed an invariant)
Solution: Take the largest set of states that will result in a $\varphi$-state when $\tau$ is taken. How?

Precondition of $\varphi$ w.r.t. $\tau$

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

a state $s$ satisfies $pre(\tau, \varphi)$ iff all $\tau$-successors of $s$ satisfy $\varphi$.

Note:
$s$ trivially satisfies $pre(\tau, \varphi)$ if it does not have any $\tau$-successors (i.e., $\tau$ is not enabled in $s$).
Precondition of \( \varphi \) w.r.t. \( \tau \) (Con’d)

Example:

\[ V: \{x\} \text{ integer} \]
\[ \rho_\tau: x > 0 \land x' = x - 1 \]
\[ \varphi: x \geq 2 \]
\[ \text{pre}(\tau, \varphi): \]
\[ \forall x'. \ x > 0 \land x' = x - 1 \rightarrow x' \geq 2 \]
\[ x > 0 \rightarrow x - 1 \geq 2 \]
\[ x \leq 0 \lor x \geq 3 \]
\[ \frac{j}{\tau} \]
\[ x \leq 0 \lor x \geq 3 \]
\[ \frac{j+1}{\tau} \]
\[ x \geq 2 \]

Properties of \( \text{pre}(\tau, \varphi) \)

By the definition of \( \text{pre}(\tau, \varphi) \),
\[ \{ \chi \land \varphi \land \text{pre}(\tau, \varphi) \} \tau \{ \varphi \} \]
is guaranteed to be state-valid.

But we have to justify adding the conjunct \( \text{pre}(\tau, \varphi) \) to the antecedent.
This can be done in two ways:
1. Incremental: prove \( \Box \text{pre}(\tau, \varphi) \)
2. Strengthening: prove \( \Box (\varphi \land \text{pre}(\tau, \varphi)) \)
Properties of $\text{pre}(\tau, \varphi)$ (Con’d)

Claim: If $\varphi$ is $P$-invariant then so is $\text{pre}(\tau, \varphi)$ for every $\tau \in T$.

Proof:
Suppose $\varphi$ is $P$-invariant, but $\text{pre}(\tau, \varphi)$ is not $P$-invariant.

Then there exists a $P$-accessible state $s$ such that $s \not\models \text{pre}(\tau, \varphi)$.

But then, by the definition of $\text{pre}(\tau, \varphi)$, there exists a $\tau$-successor $s'$ of $s$ such that $s' \not\models \varphi$.

Since $s$ is $P$-accessible, $s'$ is also $P$-accessible, contradicting that $\varphi$ is a $P$-invariant.

Properties of $\text{pre}(\tau, \varphi)$ (Con’d)

Definition: A transition $\tau$ is said to be self-disabling if for every state $s$, $\tau$ is disabled in all $\tau$-successors of $s$.

Claim: For every assertion $\varphi$ and self-disabling transition $\tau$

$$\{\varphi \land \text{pre}(\tau, \varphi)\}\, \tau\, \{\varphi \land \text{pre}(\tau, \varphi)\}$$

is state-valid.

Proof:
Assume $s \models \varphi \land \text{pre}(\tau, \varphi)$.

Then by definition of $\text{pre}(\tau, \varphi)$, for every $s'$, $\tau$-successor of $s$,
$$s' \models \varphi.$$  

Since $\tau$ is self-disabling, $\tau$ is disabled in all $\tau$-successors $s'$ of $s$, and so trivially
$$s' \not\models \text{pre}(\tau, \varphi)$$

Thus for all $\tau$-successors $s'$ of $s$,
$$s' \not\models \varphi \land \text{pre}(\tau, \varphi).$$
Heuristic

If the verification condition
\[ \{ \chi \land \varphi \} \tau \{ \varphi \} \]
is not state-valid:

Find \( \text{pre}(\tau, \varphi) \) and then

- Strengthening approach:
  strengthen \( \varphi \) by adding the conjunct \( \text{pre}(\tau, \varphi) \)
  prove \( \square(\varphi \land \text{pre}(\tau, \varphi)) \)
  or,
- Incremental approach:
  prove \( \square \text{pre}(\tau, \varphi) \)
  and add \( \text{pre}(\tau, \varphi) \) to \( \chi \).

Note:
\( \text{pre}(\tau, \varphi) \) is not guaranteed to be an inductive invariant,
so the premises of INV have to be checked again.

Example:

local \( x \): integer where \( x = 1 \)
\[
\begin{align*}
\ell_0 &: \text{request } x \\
\ell_1 &: \text{critical} \\
\ell_2 &: \text{release } x
\end{align*}
\]

We want to prove
\[
\Box (\text{at}_{-\ell_1} \rightarrow x = 0)
\]

Problem:
\[ \{ \text{at}_{-\ell_1} \rightarrow x = 0 \} \tau_{\ell_0} \{ \text{at}_{-\ell_1} \rightarrow x = 0 \} \]
is not state-valid.

If we use the above heuristic we get
\[
\text{pre}(\tau_{\ell_0}, \varphi) = \\
\forall x', \pi'. (\text{move}(\ell_0, \ell_1) \land x > 0 \land x' = x - 1) \\
\rho_{\ell_0} \rightarrow (\text{at}'_{-\ell_1} \rightarrow x' = 0)
\]
Example (Con’d):

\[
\text{pre}(\tau_{\ell_0}, \varphi) = \\
\forall x', \pi'. (\text{move}(\ell_0, \ell_1) \land x > 0 \land x' = x - 1) \\
\rho_{\ell_0} \Rightarrow (at'_{\ell_1} \rightarrow x' = 0)
\]

Since

\[
\text{move}(\ell_0, \ell_1) \rightarrow at_\ell_0 = t, at'_{\ell_1} = t
\]

\[
x' = x - 1 \land x' = 0 \rightarrow x = 1
\]

it simplifies to

\[
\text{pre}(\tau_{\ell_0}, \varphi): at_\ell_0 \land x > 0 \rightarrow x = 1
\]

Substituted form of \text{pre}(\tau, \varphi)

Many transition relations have the form

\[
\rho_\tau: C_\tau \land V' = E
\]

where \(C_\tau\) is the enabled condition of \(\tau\).

And so

\[
\text{pre}(\tau, \varphi): \forall V'. C_\tau \land V' = E \rightarrow \varphi'
\]

can be simplified to

\[
\forall V'. C_\tau \rightarrow \varphi[E/V]
\]

replacing all primed variables by its corresponding expression,

thus the quantifier can be eliminated to obtain

\[
\text{pre}(\tau, \varphi): C_\tau \rightarrow \varphi[E/V]
\]

Show that \(\varphi \land \text{pre}(\tau_{\ell_0}, \varphi)\) is inductive

("strengthening approach")

8-11 8-12
Example: Program mux-pet1 (Fig. 2.25) (Peterson's Algorithm for mutual exclusion)

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

\[
\ell_0: \text{ loop forever do} \\
| \ell_1: \text{ noncritical} \\
| \ell_2: (y_1, s) := (T, 1) \\
| \ell_3: \text{ await } (\neg y_2) \lor (s \neq 1) \\
| \ell_4: \text{ critical} \\
| \ell_5: y_1 := F
\]

\[
P_1 :: \\
| m_0: \text{ loop forever do} \\
| \text{noncritical} \\
| (y_2, s) := (T, 2) \\
| \text{await } (\neg y_1) \lor (s \neq 2) \\
| \text{critical} \\
| y_2 := F
\]

Example: Program mux-pet1 (Fig. 2.25) (Con’d)

We want to prove mutual exclusion:

\[
\neg (at_{\ell_4} \land at_{m_4})
\]

Bottom-up invariants:

\[
\begin{align*}
\varphi_0: & \quad s = 1 \lor s = 2 \\
\varphi_1: & \quad y_1 \leftrightarrow at_{\ell_3..5} \\
\varphi_2: & \quad y_2 \leftrightarrow at_{m_3..5}
\end{align*}
\]

Problem: the verification conditions

\[
\{\varphi_0 \land \varphi_1 \land \varphi_2 \land \psi\} \ell_3 \{\psi\} \\
\{\varphi_0 \land \varphi_1 \land \varphi_2 \land \psi\} m_3 \{\psi\}
\]

are not state-valid
Example: Program mux-pet1 (Fig. 2.25) (Con’d)

\[ \text{pre}(\tau_{l3}, \psi): \forall \pi': \left( \text{move}(l_3, l_4) \land (\neg y_2 \lor s \neq 1) \right) \rightarrow \neg(\text{at}_{l_4} \land \text{at}_{m4}) \]

since

\[ \text{move}(l_3, l_4) \text{ implies } \text{at}_{l_4} = 1, \text{ at}_{m4} = \text{at}_{m4} \]

\[ \text{pre}(\tau_{l3}, \psi) \text{ simplifies to:} \]

\[ at_{-l_3} \land (\neg y_2 \lor s \neq 1) \rightarrow \neg at_{-m4} \]

\[ \varphi_3: at_{-l_3} \land at_{-m4} \rightarrow y_2 \land s = 1 \]

\[ \text{pre}(\tau_{m3}, \psi): \forall \pi' \ldots \ldots \]

simplifies to:

\[ \varphi_4: at_{-l_4} \land at_{-m3} \rightarrow y_1 \land s = 2 \]

Show that \( \varphi_3: \text{pre}(\tau_{l3}, \psi) \) and \( \varphi_4: \text{pre}(\tau_{m3}, \psi) \) are inductive relative to \( \varphi_0 \land \varphi_1 \land \varphi_2 \) ("incremental approach")

Then show that \( \psi \) is inductive relative to \( \varphi_0 \land \ldots \land \varphi_4 \).
Example: $pre$ may never terminate

The transition is
\[ \rho_\tau : x' = x + y \land y' = y \]

The property is
\[ \varphi : x \geq 0 \]

The VC is
\[ \frac{x' = x + y \land y' = y}{\rho_\tau} \frac{x \geq 0}{\varphi} \frac{x' \geq 0}{\varphi'} \]

which is not state valid.

Step 1: The precondition is
\[ pre(\tau, x \geq 0) : \forall x', y' : x' = x + y \land y' = y \rightarrow x' \geq 0 \]

that is $y \geq -x$.

Attempting to prove $\Box pre(\tau, \varphi)$ state valid, the VC
\[ \frac{x' = x + y \land y' = y}{\rho_\tau} \frac{x \geq 0}{\varphi} \frac{y' \geq -x'}{pre'} \]

is not state-valid.

Step 2: Compute $pre(\tau, y \geq -x)$

\[ \forall x', y' : x' = x + y \land y' = y \rightarrow y' \geq -x' \]

that is $y \geq -\frac{x}{2}$.

In general the precondition
\[ pre(\tau, y \geq -\frac{x}{n}) : y \geq -\frac{x}{n + 1} \]

Taking the limit as $n$ approaches infinity, we obtain
\[ y \geq 0 \]

which is what we want.
Finite-State Algorithmic Verification

finite-state program $P$

each $x \in V$ assumes only finitely many
values in all $P$-computations

Therefore,
there are only finitely many distinct
$P$-accessible states.

Example:
MUX-PET1 (Fig 2.25) is finite-state program:
$s = 1, 2$
$y_1 = T, F \quad y_2 = T, F$
$\pi$ can assume at most 36 different values

Example: Program mux-pet1 (Fig. 2.25)
(Peterson’s Algorithm for mutual exclusion)

local $y_1, y_2$: boolean where $y_1 = F, y_2 = F$
$s$: integer where $s = 1$

$\ell_0$: loop forever do
\begin{align*}
\ell_1 & : \text{noncritical} \\
\ell_2 & : (y_1, s) := (T, 1) \\
\ell_3 & : \text{await } (\neg y_2) \lor (s \neq 1) \\
\ell_4 & : \text{critical} \\
\ell_5 & : y_1 := F
\end{align*}

$P_1 ::$

$m_0$: loop forever do
\begin{align*}
m_1 & : \text{noncritical} \\
m_2 & : (y_2, s) := (T, 2) \\
m_3 & : \text{await } (\neg y_1) \lor (s \neq 2) \\
m_4 & : \text{critical} \\
m_5 & : y_2 := F
\end{align*}

$P_2 ::$
Algorithm (transition-graph)

For a given finite-state program $P$,
Incrementally construct the state-transition graph $G_P$, where each node represents a state.

- **Initially**
  Place as nodes in $G_P$ all initial states (satisfy $\Theta$)

- **Repeat** until no new nodes or new edges can be added to $G_P$
  
    For some $s \in G_P$, let $s_1, \ldots, s_k$ be its successors
    
    Add to $G_P$ all new nodes in $\{s_1, \ldots, s_k\}$
    
    and draw edges connecting $s$ to $s_i$, $i = 1, \ldots, k$

Algorithmic Verification of Invariance

For assertion $q$,
To check validity of $\Box q$ over finite-state program $P$:

1. Construct the state-transition graph $G_P$.

2. Check if $q$ holds in each state of the graph.

**Example:** Program MUX-SEM (Fig 2.26)

Generates finite state-transition graph (Fig 2.27)

Check assertion

$$\varphi: \neg(at_{-\ell_3} \land at_{-m_3})$$

in the graph.

$\varphi$ holds over all accessible states.
Thus, $\Box \varphi$ for MUX-SEM.
Program MUX-SEM (Fig. 2.26)
(mutual exclusion by semaphores)

local $y$: integer where $y = 1$

$P_1:: \begin{bmatrix} \ell_0: \text{loop forever do} \\ \ell_1: \text{noncritical} \\ \ell_2: \text{request } y \\ \ell_3: \text{critical} \\ \ell_4: \text{release } y \end{bmatrix}$ $|| P_2:: \begin{bmatrix} m_0: \text{loop forever do} \\ m_1: \text{noncritical} \\ m_2: \text{request } y \\ m_3: \text{critical} \\ m_4: \text{release } y \end{bmatrix}$
Example: Program MUX-PET1 (Fig. 2.25)

State-transition graph $G_P$ (Fig. 2.28)

$(i, j, v)$ means $\pi: \{\ell_i, m_j\}, \ s: v$

No $y_1, y_2$ since

$y_1 = T \text{ iff } 3 \leq i \leq 5$

$y_2 = T \text{ iff } 3 \leq j \leq 5$

Property checked

$\not\psi (\not at_{\ell_4} \land at_{m_4})$

Example: Program mux-pet1(Fig. 2.25)
(Peterson’s Algorithm for mutual exclusion)

local $y_1, y_2$: boolean where $y_1 = F, y_2 = F$

$s$: integer where $s = 1$

$\ell_0$: loop forever do

$P_1 :: \begin{array}{l}
\ell_1: \text{noncritical} \\
\ell_2: (y_1, s) := (T, 1) \\
\ell_3: \text{await } (\neg y_2) \lor (s \neq 1) \\
\ell_4: \text{critical} \\
\ell_5: y_1 := F
\end{array}$

$\begin{array}{ll}
\hline
m_0: \text{loop forever do} \\
\hline
m_1: \text{noncritical} \\
m_2: (y_2, s) := (T, 2) \\
m_3: \text{await } (\neg y_1) \lor (s \neq 2) \\
m_4: \text{critical} \\
m_5: y_2 := F
\end{array}$

$P_2 :: \begin{array}{l}
\hline
\end{array}$
Completeness of rule INV

Rule INV (general invariance)

For assertions $\varphi$, $q$,

- I1. $\vdash \varphi \rightarrow q$
- I2. $\vdash \Theta \rightarrow \varphi$
- I3. $\vdash \{\varphi\} \land \{\varphi\}$

$\vdash \Box q$

Theorem (Relative completeness of rule INV)

For every assertion $q$ such that

$\Box q$ is $P$-valid

there exists an assertion $\varphi$ such that I1 – I3 are provable from state validities
We actually show
“completeness relative to
first-order reasoning”
taking all state-valid assertions as axioms

Outline of proof

Given FTS $P$ with system variables (program + control variables)

$$\bar{y} = (y_1, \ldots, y_m)$$

- Assume $\Box q$ is $P$-valid, i.e.,
  ($\dagger$) $q$ holds over every $P$-accessible state

- Construct (to be shown) accessibility assertion
  $acc_P(\bar{y})$
  such that for any state $s$,
  ($*$) $s$ is $P$-accessible state \iff $s \not\models acc_P$

- Take $\varphi = acc_P$

We have to show:
1. $acc_P$ satisfies I1 – I3
2. $acc_P$ can be “constructed”

1. $acc_P$ satisfies I1 – I3

- Premise I1: $\varphi \rightarrow q$
  $$s \not\models acc_P \quad (\Rightarrow) \quad s \text{ is } P\text{-accessible state}$$
  $$\Rightarrow \quad s \not\models q$$
  Thus
  $$\varphi \rightarrow q$$
  is state-valid

- Premise I2: $\Theta \rightarrow acc_P$
  $$s \not\models \Theta \quad (\Rightarrow) \quad s \text{ is } P\text{-accessible}$$
  $$\Rightarrow \quad s \not\models acc_P$$
  Thus
  $$\Theta \rightarrow acc_P$$
  is state-valid
Premise I3: for every $\tau \in T$,
$$\rho_\tau \land acc_P \rightarrow acc'_P,$$
where $acc'_P = acc_P(y')$.

Take $s'$ to be a $y$-variant of $s$ ($s$ agrees with $s'$ on all variables other than $\overline{y}$) and for each $y_i$ take
$$s'[y_i] = s[y'_i].$$

Then
$$s \not\models \rho_\tau \Rightarrow s' \text{ is a } \tau\text{-successor of } s$$
$$s \not\models acc_P \quad \overset{(\ast)}{\Rightarrow} \quad s \text{ is } P\text{-accessible} \quad \overset{(\ast)}{\Rightarrow} \quad s' \text{ is } P\text{-accessible}$$
$$s' \not\models acc_P$$
$$\Rightarrow \quad s \not\models acc'_P$$

Example:
$V: \{y\}$ $\Theta: y = 0$
$T: \{\tau_I, \tau\}$, where $\rho_\tau: y' = y + 2$
For this program: $acc_P(y): y \geq 0 \land even(y)$

2. Construction of $acc_P$

Assume assertion language includes dynamic array $a$ over $D$

Array $a$ is viewed as function,
$$a: [1..n] \mapsto D$$
where $n$ is the size of the array

The assumption is not essential
We can use Gödel numbering
$$(k_1, \ldots, k_n) \mapsto n = p_1^{k_1} \cdots p_n^{k_n}$$
where $p_i$ is the $i$th prime number
Case: single-variable \( y \)

Define

\[
\text{acc}_P(y): \ (\exists n > 0) \ (\exists a \in [1..n] \mapsto D).
\]

\( \text{init} \land \text{last} \land \text{evolve} \)

where

\( \text{init}: \ \Theta(a[1]) \)
\( \text{last}: \ a[n] = y \)
\( \text{evolve}: \ \forall i \cdot 1 \leq i < n. \ \bigvee_{\tau \in T} \rho_{\tau}(a[i], a[i+1]) \)

i.e., there exists an array \( a \), such that

- \( a[1] \) is an initial state
- \( a[n] \) has value \( y \) (last element)
- every two consecutive elements are related by some transition relation

array \( a \) represents a prefix

\( s_1, \ldots, s_n \)

of a computation where \( a[i] \) stands for

the value of \( y \) at state \( s_i \)

Claim:

For any value \( d \in D \),

\[ \text{acc}_P(d) = T \]

iff

\( d \) is a possible value of \( y \) in a \( P \)-accessible state

\( \text{acc}_P(d) \) asserts the existence of a computation prefix

that leads to a state where \( y = d \).
Example: Transition system EVEN

\[ V: \{y\} \quad \text{ranges over } \mathbb{Z} \text{ (the integers)} \]
\[ \Theta: \ y = 0 \]
\[ \rho_{\tau}: \ y' = y + 2 \]

\[ acc_P(y): \]
\[ (\exists n > 0)(\exists a \in [1..n] \mapsto \mathbb{Z}). \]
\[ \left( a[1] = 0 \land a[n] = y \land \right) \]
\[ \left( \forall i . 1 \leq i < n . a[i + 1] = a[i] + 2 \right) \]

simplifies to

\[ (\exists n > 0)(\exists a \in [1..n] \mapsto \mathbb{Z}). \]
\[ \left( a[n] = y \land \right) \]
\[ \left( \forall i . 1 \leq i \leq n . a[i] = 2 \cdot (i - 1) \right) \]

simplifies to

\[ y \geq 0 \land \text{even}(y) \]

Precisely characterizes the values that \( y \) may assume in \( P \)-accessible states of EVEN

Discussion

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

The shaded area is preserved by all transitions. Its description is much simpler than that of \( acc_P \).
Multivariable $\mathbf{y} = (y_1, \ldots, y_m)$ case

Use 2-dimensional array $\mathbf{a}$

\[
\begin{array}{cccc}
  y_1 & \cdots & y_m \\
  a[1, 1] & \cdots & a[1, m] \\
  a[2, 1] & \cdots & a[2, m] \\
  \vdots & \ddots & \vdots \\
  \vdots & \ddots & \vdots \\
\end{array}
\]

Example: Transition system FACT

$y, z$ ranging over $\mathbb{N}$ (the nonnegative integers)

$\Theta: y = 1 \land z = 1$

$\rho, \tau: y' = y + 1 \land z' = (y + 1) \cdot z$

Construction of $\text{acc}_P$:

$(\exists n > 0)(\exists a \in [1..n] \times [1, 2] \mapsto \mathbb{N}).$

\[
\begin{pmatrix}
  a[1, 1] = 1 \land a[1, 2] = 1 \\
  a[n, 1] = y \land a[n, 2] = z \\
  \land \\
  \forall i: 1 \leq i < n: a[i + 1, 1] = a[i, 1] + 1 \land \\
  a[i + 1, 2] = (a[i, 1] + 1) \cdot a[i, 2]
\end{pmatrix}
\]
\((\exists n > 0)(\exists a \in [1..n] \times [1, 2] \mapsto \mathbb{N})\).

\[
\begin{pmatrix}
a[1, 1] = 1 & a[1, 2] = 1 \\
a[n, 1] = y & a[n, 2] = z \\
\forall i: 1 \leq i < n: a[i + 1, 1] = a[i, 1] + 1 & a[i + 1, 2] = (a[i, 1] + 1) \cdot a[i, 2]
\end{pmatrix}
\]

simplifies to

\((\exists n > 0)(\exists a \in [1..n] \times [1, 2] \mapsto \mathbb{N})\).

\[
\begin{pmatrix}
a[n, 1] = y & a[n, 2] = z \\
\forall i: 1 \leq i \leq n: a[i, 1] = i & a[i, 2] = i!
\end{pmatrix}
\]

simplifies to

\[y \geq 1 \land z = y!\]

Precisely characterizes the \(P\)-accessible states
for the transition system FACT

8-39