CS 357D

Lecture 2

Computational Model

Properties of interest

Invariants: overapproximation of the reachable state space

Loop termination: demonstrated by the existence of a ranking function

Computational Model

Behaviors: sequences of states

System description: state transition systems
compact first-order representation of all sequences of states that can be generated by a system

Programming language: SPL (simple programming language)
with well-defined semantics in terms of transition systems

Reference:

Semantics

Java program
C++ program

Java byte code SPL program Petri Net Assembly Code

Transition system

sequences of states

Conditions on program text
Conditions on description
Properties of interest
Semantics

System Description: Transition systems

Set of typed variables
Example: \{x:int, y:int\}

Initial condition:
first-order formula
Example: \(x=0 \land y=0\)

Compact first-order representation of all sequences of states that can be generated by a system

States

V: Vocabulary -- set of typed variables \(\{x,y: \text{integer}, b: \text{boolean}\}\)
expression over V 
assertion over V

s: state -- interpretation of all variables \(\{x:2,y:3,b: \text{true}\}\)
extends to expressions and assertions

\(s[x+y]=5\)
\(s[x\cdot y]=\text{false}\)

\(\Sigma: \text{set of all states}\) \(\mathbb{Z} \times \mathbb{Z} \times \{\text{true, false}\}\)

Transitions

\(\mathcal{T}: \text{finite set of transitions}\)

Example:
\(T(x+2) = \{x:3, x:4\}\)
\(T(x+3) = \{x:4, x:5\}\)
\(T(x:4) = \{x:5, x:6\}\)
\(T(x:5) = \{x:6, x:7\}\)
\(T(x:6) = \{x:7, x:8\}\)

represented by a transition relation \(\rho_T(V,V')\)

\(V: \text{values of variables in the current state}\)
\(V': \text{values of variables in the next state}\)

\(\rho_T: x'=x+1 \lor x'=x+2\)
A transition $\tau$ is **enabled** in a state $s$ if: $\tau(s) \neq \emptyset$

A transition $\tau$ is **disabled** in a state $s$ if: $\tau(s) = \emptyset$

**Example:**

Transition $\tau$ with $\rho_\tau: (x = 0 \lor x = 1) \land (x' = x + 1 \lor x' = x + 2)$

$\tau(x:0) = \{x:1, x:2\}$

$\tau(x:1) = \emptyset$

$\tau(x:2) = \emptyset$

$\tau(x:3) = \emptyset$

Infinite sequence of states

$\sigma$: $s_0 \ s_1 \ s_2 \ s_3 \ s_4 \ \ldots$

is a **run** of $\Phi$ if

- **Initiality:** $s_0 \models \Theta$  \hspace{1cm} ($s_0$ is an initial state)
- **Consecutive:** for all $i > 0$

  $s_{i+1}$ is a $\tau$-successor of $s_i$

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

---

**Runs: Example**

$V$: \{x:integer\}

$\Theta$: $x=0$

$\mathcal{T}$: \{T_1, T_2, T_3\} with

\[
\begin{align*}
\rho_{T_1}: & \quad x' = x + 1 \lor x' = x + 3 \\
\rho_{T_2}: & \quad x' = x + 2 \lor x' = 2x \\
\rho_{T_3}: & \quad x' = x
\end{align*}
\]

$\sigma_1$: 0, 1, 2, 3, 4, 5, 6, 7, \ldots

$\sigma_2$: 0, 0, 0, 0, 0, 0, 0, \ldots

$\sigma_3$: 0, 2, 4, 8, 16, 19, \ldots

$\sigma_4$: 0, 1, 1, 3, 3, 5, 5, 7, \ldots

$\sigma_5$: 0, 1, 2, 3, 5, 6, 8, 9, 18, \ldots

---

**Runs: Example**

$V$: \{x:integer\}

$\Theta$: $x=0$

$\mathcal{T}$: \{T_1, T_2, T_3\} with

\[
\begin{align*}
\rho_{T_1}: & \quad (x=0 \lor x=1) \land (x'=x+1 \lor x'=x+3) \\
\rho_{T_2}: & \quad x'=x+2 \lor x'=2x \\
\rho_{T_3}: & \quad x'=x
\end{align*}
\]

$\sigma_1$: 0, 1, 2, 3, 4, 5, 6, 7, \ldots

$\sigma_2$: 0, 0, 0, 0, 0, 0, 0, \ldots

$\sigma_3$: 0, 2, 4, 8, 16, 32, \ldots

$\sigma_4$: 0, 1, 1, 3, 3, 5, 5, 7, \ldots

$\sigma_5$: 0, 1, 2, 3, 5, 6, 8, 9, 18, \ldots

not a run!
System Description: Summary

Transition system: \( \Phi: < V, \Theta, \mathcal{T} > \)

Run: Initiality + Consecution

\( \mathcal{L}(\Phi) \): all runs of \( \Phi \)  
“Behavior of the program”

(all sequences of states that satisfy Initiality and Consecution)

SPL

Simple programming language with constructs (a.o.):

- assignment
- conditional (if - then - else)
- concatenation
- selection
- while

Static global variable initialization

Statements are labeled
- define program locations (equivalence relation on labels)

Semantics

Given an SPL program \( P \) we can construct the corresponding transition system \( \Phi: < V, \Theta, \mathcal{T} > \).

- each program statement corresponds to a transition
  - no sequential structure in transition systems, therefore control is modeled explicitly by a control variable \( \pi \) that ranges over program locations

- \( V \): program variables \( \cup \{ \pi \} \)
- \( \Theta \): program initial condition
SPL statements

assignment statement
\[ l_1: x := e; l_2 \]
translates into transition \( \tau \) with transition relation
\[ \rho_\tau: \pi_1 \land x' = e \land \pi_2 \land \text{pres}(V - \{x,\pi\}) \]
conditional statement
\[ l_1: \text{if } c \text{ then } l_2; S_1 \text{ else } l_3; S_2 \]
translates into transition \( \tau \) with transition relation
\[ \rho_\tau: \pi_1 \land ((c \land \pi_1 = l_2) \lor (\neg c \land \pi_1 = l_3)) \land \text{pres}(V - \{\pi\}) \]

SPL semantics

Full semantics of SPL in

Reachable state space

state \( s \) is \( \Phi \)-reachable if it appears in some \( \Phi \)-run
\[ \sigma: S_0, S_1, S_2, S_3, S_4 \ldots \]

system \( \Phi \) is finite-state if the set of \( \Phi \)-reachable states is finite

Notation:
\[ \Sigma: \text{ state space} \]
\[ \Sigma_{\Phi^-}: \Phi \text{-reachable state space} \]

Example:
\[ V: \{b_1, b_2\} \]
\[ \Theta: b_1 \land b_2 \]
\[ \mathcal{J} \subset: \{ \tau \} \text{ with } \rho_\tau: b_1 \land b_2 \land \neg b_2 \]

Properties of interest
conditions on program text
conditions on description
sequences of states
Reachable state space

state s is \( \Phi\)-reachable if it appears in some \( \Phi\)-run

\[ \sigma: S_0 \quad S_1 \quad S_2 \quad S_3 \quad S_4 \quad ............. \]

system \( \Phi \) is finite-state if the set of \( \Phi\)-reachable states is finite

Notation: \( \Sigma \) : state space
\( \Sigma_{\Phi} \) : \( \Phi\)-reachable state space

Example:
\( \Sigma = \mathbb{N} \)
\( \nu: \{x: \text{int}\} \)
\( \Theta: x=0 \)
\( \mathcal{F}_\nu: \{ \tau \} \) with \( \rho: x=0 \wedge x'=x+1 \)

Invariants

An invariant \( q \) of program \( P \) is

- a superset of the reachable state space of \( P \)
- \( q \) is an assertion (first-order formula)
- also written:

\[ P \models q \quad \text{all reachable states of } P \text{ satisfy } q \]
\[ P \models \square q \quad \text{all states of all runs of } P \text{ satisfy } q \]
Properties of program behaviors

<table>
<thead>
<tr>
<th>First-order logic</th>
<th>Temporal logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>models are states</td>
<td>models are sequences of states</td>
</tr>
</tbody>
</table>

Assertion language $\mathcal{A}$:

- first-order language over system variables (+ theories for their domains)

Formulas in $\mathcal{A}$: state formulas (aka assertions)

- evaluated over a single state

  $$s \models p \iff s[p] = \text{true}$$

Examples:
- $s: <x:4, y:1>$
- $s \models x=0 \lor y=1$
- $s \models x > y$
- $s \models x = y+3$
- $s \models \text{odd}(x)$

Specification: underlying assertion language

Assertions represent sets of states

- assertion $p$ represents the set of states for which $p$ is true
- temporal formula $\varphi$ represents the set of sequences of states for which $\varphi$ is true

Example:
- $\Sigma = \{x > 0, x > 5, x^2 < 100\}$

Specification: underlying assertion language

- assertion $p$ is state-satisfiable if $s \models p$ for some state $s \in \Sigma$
- Example: $x > 0$

- assertion $p$ is state-valid if $s \models p$ for all states $s \in \Sigma$
- Example: $x > y \rightarrow x + 1 > y$
State validity and system state validity

Given a system $\Phi$

- for a state formula $p$
  $\models p$
  $p$ holds in all states
  $p$ is state-valid

Example: $\models x=1 \rightarrow x>0$

- for a state formula $q$
  $\Phi \models q$
  $q$ holds in all $\Phi$-reachable states
  $q$ is $\Phi$-state-valid

Example: $\models x=1 \rightarrow x>0$

Verification conditions: examples

$\{p\} \tau \{q\}$

$\{x>0\} x'=x+1 \{x>1\}$
$x>0 \land x'=x+1 \rightarrow x'>1$
substitute $x+1$ for $x'$: $x>0 \rightarrow (x+1) > 1$

$\{x>0\} x'=x+1 \{\text{true}\}$
$x>0 \land x'=x+1 \rightarrow \text{true}$

$\{x\geq 0\} x>0 \land x'=x-1 \{x\geq 0\}$
$\{\text{true}\} x>0 \land x'=x-1 \{\geq 0\}$

Verification condition

Starting from a state that satisfies $p$, transition $\tau$ leads to a state that satisfies $q$

aka "Hoare triple" $\{p\} \tau \{q\}$

Proving invariance properties

Invariant: $\square p$ for state formula $p$

We want to prove $\Phi \models \square p$
  every state of every run of $\Phi$ satisfies $p$

Recall: A sequence of states $\sigma$: $s_0,s_1,s_2,...$
is a run of $\Phi$: $<V,\Theta,\mathcal{T}>$ if

- Initiality: $s_0 \models \Theta$
- Consecution: for each $j \geq 0$, $s_{j+1}$ is a $\tau$-successor of $s_j$,
  for some $\tau \in \mathcal{T}$
Proving invariance properties

Proving $\Phi \models \Box p$

means proving that every state of every sequence of states that satisfies

- **Initiality:** $s_0 \models \Theta$
- **Consecution:** for each $j \geq 0$, $s_{j+1}$ is a $\tau$-successor of $s_j$,
  for some $\tau \in \mathcal{T}$

also satisfies $p$

Proof by induction:

- **Base case:** $\Theta \rightarrow p$ ensures that every initial state satisfies $p$
- **Inductive step:** $p \land \rho_{\tau} \rightarrow p'$ for every $\tau \in \mathcal{T}$
  ensures that $p$ is preserved by all transitions

Verification rule B-INV (basic invariance)

For assertion $q$

\[
\begin{align*}
B1. & \quad \Phi \models \Theta \rightarrow q \\
B2. & \quad \Phi \models \{q\} \mathcal{T} \{q\} \\
\end{align*}
\]

$\{q\} \mathcal{T} \{q\}$ stands for $\{q\} T \{q\}$ for all $\tau \in \mathcal{T}$

B-INV reduces the proof of an invariant to checking the validity of

$\models \mathcal{T} \chi + 1$ first-order verification conditions in the underlying assertion language.

Semantics

<table>
<thead>
<tr>
<th>Java program</th>
<th>C++ program</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

Transition system

- **Properties of interest**
- **for invariants**

**B-INV : example**

$\Phi$: to prove $\Phi \models \Box(x \geq 0)$

\[\begin{align*}
V: \{x\} \\
\Theta: x=0 \\
\end{align*}\]

$\mathcal{T}: \{T_1, T_2\}$ with $\rho_{T_1}: x'=x+1$

$\rho_{T_2}: x \geq 0 \land x'=x-1$

\[
\begin{align*}
B1: \quad x=0 & \rightarrow x \geq 0 \quad \checkmark \\
B2: \quad x \geq 0 \land x'=x+1 & \rightarrow x \geq 0 \quad \checkmark \\
\end{align*}
\]

\[
\begin{align*}
B1. & \quad \Phi \models \Theta \rightarrow q \\
B2. & \quad \Phi \models \{q\} T_1 \{q\} \\
B2. & \quad \Phi \models \{q\} T_2 \{q\} \\
\end{align*}
\]
**B-INV : example**

\[ \phi: \quad \text{to prove} \quad \phi \models \Box(x \geq 0) \]

\[
\begin{align*}
V & : \{x,y\} \\
\Theta & : x=0 \land y=0 \\
\mathcal{J} & : \{T_1,T_2\} \text{ with } \rho_{T_1}: x'=x+y \land y'=y+1 \\
& \quad \rho_{T_2}: x>0 \land x'=x-1
\end{align*}
\]

- **B1**: \( x=0 \land y=0 \rightarrow x \geq 0 \) \( \checkmark \)
- **B2**: \( x \geq 0 \land x'=x+y \land y'=y+1 \rightarrow x' \geq 0 \) \( \times \)
  \[
  x \geq 0 \land x>0 \land x'=x-1 \rightarrow x' \geq 0 \quad \checkmark
  \]

\( x \geq 0 \) is an invariant, but it is not **inductive**

---

**Verification rule B-INV (basic invariance)**

For assertion \( q \)

- B1. \( \phi \models \Theta \rightarrow q \)
- B2. \( \phi \models \{q\} \mathcal{J} \{q\} \)

\( \phi \models \Box q \)

if B1 and B2 are (state) valid then \( q \) is inductive

**every inductive assertion is an invariant**

the converse is not true: not every invariant is inductive

---

**Non-inductive invariants**

**Strategy:** strengthen \( q \) until it is inductive
### Strategy 1: Strengthening

Let: \( \Phi: \{x, y\} \)
- \( \Theta: x = 0 \land y = 0 \)
- \( \mathcal{E}: \{T_1, T_2\} \) with \( \rho_{T_1}: x' = x + y \land y' = y + 1 \)
- \( \rho_{T_2}: x > 0 \land x' = x - 1 \land y' = y \)

B1: \( x = 0 \land y = 0 \rightarrow x \geq 0 \land y \geq 0 \)
B2: \( x \geq 0 \land y \geq 0 \land x' = x + y \land y' = y + 1 \rightarrow x' \geq 0 \land y' \geq 0 \)
\( x \geq 0 \land y \geq 0 \) is an invariant and is inductive

### Strategy 2: Incremental Proof

Let: \( \Phi: \{x, y\} \)
- \( \Theta: x = 0 \land y = 0 \)
- \( \mathcal{E}: \{T_1, T_2\} \) with \( \rho_{T_1}: x' = x + y \land y' = y + 1 \)
- \( \rho_{T_2}: x > 0 \land x' = x - 1 \land y' = y \)

B1: \( x = 0 \land y = 0 \rightarrow x \geq 0 \)
B2: \( x \geq 0 \land y \geq 0 \land x' = x + y \land y' = y + 1 \rightarrow x' \geq 0 \)
\( x \geq 0 \) is an invariant and is inductive relative to \( y \geq 0 \)