Outline

→ Verification
→ Examples
→ Discrete transition System
  → Executions behaviors
  → Invariant
  → Proving / Verifying Invariant

The verification problem

A

CPS model A

Verification algorithm

Proof that A meets R

Counterexample showing A violates R

Requirements R

Software Machines

Diff Equs

Markov Chains

Hybrid Systems

All behaviors of the model A satisfies the requirement

Find particular behavior of A that violates the requirement
Example 1 (from last lecture)
Two vehicles on a single lane highway

- $a_f[t] = a_{min}$
- $v_f[t + 1] = v_f[t] + a_f[t] \Delta$
- $c_f[t + 1] = c_f[t] + v_f[t] \Delta + \frac{1}{2} a_f[t] \Delta^2$

- $a_r[t] \in [a_{min}, a_{max}]$
- $v_r[t + 1] = v_r[t] + a_r[t] \Delta$
- $c_r[t + 1] = c_r[t] + v_r[t] \Delta + \frac{1}{2} a_r[t] \Delta^2$

$Q_o = \{ c_f[0] = c_r[0] + d_o, v_r[0] = v_0 \}$

$A = \{ (a_f, a_r) \mid a_f, a_r \in [a_{min}, a_{max}] \}$

$D \subseteq Q \times A \times Q \quad q \xrightarrow{(a_f, a_r)} q'$ iff

- $q' . v_f = q . v_f + a_f . \Delta \quad \Delta$ time elapsing
- $q' . v_r = q . v_r + a_r . \Delta \quad \Delta$ time elapsing
- $q' . c_f = -$ - $-
- $q' . c_r = -$ - $-$
Why nondeterminism in the model?

Non-deterministic mechanism for Uncertainty

Sources of Uncertainty:
- Reaction Times
- Sensor Noise Range

\[ x, a \]

\[ x \xrightarrow{a} x' \]

\[ x' = x + a \Delta + \varepsilon \quad \text{where} \quad \varepsilon \in [4,7] \]
State Machine / Discrete transition System / Automaton

\[ A = \langle G, \Sigma_0, \Sigma, \Delta \rangle \]

\[ G : \text{ Set of States of the automaton} \]
\[ G \text{ is specified by a set of state variables } X \]
\[ \begin{align*}
\text{Ex} \quad X &= \{c_f, \nu_f, a_f, c_r, \nu_r, a_r, q_0, q_1, \ldots \} \\
\text{type}(c_f) &= \mathbb{R} \\
\text{type}(a_f) &= \mathbb{R} \quad \text{or} \quad \text{type}(a_f) \in [\text{amin}, \text{amax}] 
\end{align*} \]

\[ \Sigma_0 \subseteq G \text{ Set of initial states } \]
\[ |\Sigma_0| \text{ not necessarily } = 1 \]

\[ \Delta : \text{ Set of actions or labels} \]
\[ \text{ give names to how the states change} \]

\[ \Delta \subseteq G \times \Sigma \times G \quad (q, a, q') \in \Delta \text{ we write } \]
\[ 0. \quad \text{prestate } s \xrightarrow{a} q' \xleftarrow{\text{post state}} \]

\[ \exists q_1, q_2 \quad q_1 \neq q_2 \quad \text{or } q \xrightarrow{a} q_1 \land q \xrightarrow{a} q_2 \]

A can be non-deterministic
A is said to be deterministic if
\[ \forall q \in Q \forall a \forall q_1, q_2 \in Q \]
\[ \text{if } q \xrightarrow{a} q_1 \land q \xrightarrow{a} q_2 \Rightarrow q_1 = q_2 \]

post(q) \rightarrow \quad \text{post} : Q \rightarrow 2^Q

\[ \forall q' \in Q \mid \exists a \in A \quad q \xrightarrow{a} q' \]

\[ \text{Post}(Q') = \bigcup_{q' \in Q'} \text{Post}(q) \]

---

**Representation of States**

*How to represent states (in programs)*

- List \( \emptyset, \ldots, T \)
- \( |Q| \text{ finite} \quad \text{Finite State Machine FSM} \)
- \( |Q| \sim 2^{100} \)

Explicit representation will blow-up

\[ x_1, x_2, \ldots, x_{100} \in \{0, 1\} \]

bit vector

Write a formula involving the state variable

\[ \phi = x_1 \lor x_2 \subseteq \{<0,1>, <1,0>, <1,1>\} \]

\[ \bigwedge \phi \text{ represents the set of states} \]
\[ S_\phi' = \{ x \mid x \text{ satisfies } \phi \} \]

\[ S_{\phi_1} \cup S_{\phi_2} \]
\[ \phi \lor \phi_2 \]
\[ \phi_2 \land \phi_2 \]
Example 2. Microwave State machine

\[ I = \{1, 2, 3, 6, 4, 7\} \]

\[ = ! \text{Error} \]

\[ G = \{1, 2, ..., 7\} \]

\[ G_0 = \{1\} \]

\[ A = \{ \text{Start oven, Reset, Warm up}\} \]

\[ \langle 1, \text{Reset}, 5 \rangle \in D \]

\[ \langle 1, \text{Reset}, 6 \rangle \notin D \]

\[ \langle 1, \text{Close door}, 3 \rangle \in D \]

\[ D = \{ \langle 1, \text{Close door}, 3 \rangle, \langle 1, \text{Start oven}, 2 \rangle \} \]
What are behaviors of an automaton?

\[ A = \langle Q, Q_0, A, D \rangle \]

Executions model a particular run of the automaton \( A \).

An execution of \( A \) is an alternating sequence of states and actions

\[ \alpha = q_0 a_1 q_1 a_2 q_2 a_3 \ldots \]

If \( \alpha \) is finite

\[ q_0 a_1 q_1 a_2 q_2 \ldots q_k \]

Then

1. \( q_i \in Q \) \( \forall i \), \( a_i \in A \) \( \forall i \)
2. \( q_i \xrightarrow{a_{i+1}} q_{i+1} \)
3. \( q_0 \in Q_0 \)

Example: Start oven, 2, close door, 5, ...

Example 1.

\[ \alpha = x_0(a_f, q_r) x_1(a_f a_r) \ldots \]

Nondeterministic

Set of all possible executions of the system

Example 1.
A requirement \( R \) is just a subset of \( \text{Execs}_A \) \( R \subseteq \text{Execs}_A \).

Sometimes in order to check properties \( A \) we with consider requirements specified in terms of a different idealized model \( A' \).

A state \( q \in \mathcal{Q} \) is reachable if:

\[ \exists x \in \text{Exec}_A \, x.lstate = q \]

\[ \text{Reach}_A = \{ q \mid \exists x \in \text{Exec}_A \, x.lstate = q \} \]

Requirements

Common type of requirement

- Something is always true
- Something never happens

Invariant

Def. An invariant requirement is specified as a set of states such that \( \forall x \in \text{Exec}_A \, i \in I \)
alternatively  \[ \text{Reach}_A \subseteq I. \]

\[ \exists! t \text{ if we choose } C_g[t] - C_r[t] > d_0 \]
and \[ \forall t \forall C_g[t], C_r[t] > 0 \]
then
\[ \forall t \left[ C_g[t] - C_r[t] > 0 \right] \]
invariant.

Set Marked := \{\}
Queue Q := S
Marked := Marked U S
while Q is not empty
  t \leftarrow Q.dequeue()
  if t \in T return “yes”
  for each (t,u) \in E
    If u \notin Marked then
      Marked := Marked U \{u\}
      Q := enqueue(Q, u)
  return “no”