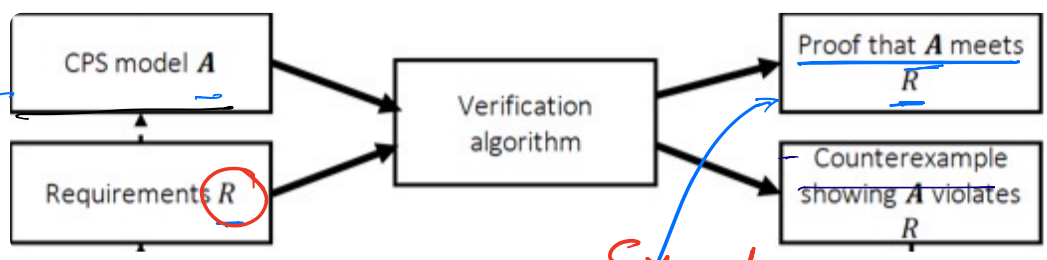
## Outline

- → Verification
- → Examples ✓
- → Discrete transition System ✓
  - ↳ Executions behaviors ✓
  - ↳ Invariant ✓
  - ↳ Proving / verifying Invariants ←

## The verification problem

$\mathcal{A}$




State Machines ←
Diff Eqns
Markov chains
Hybrid Systems

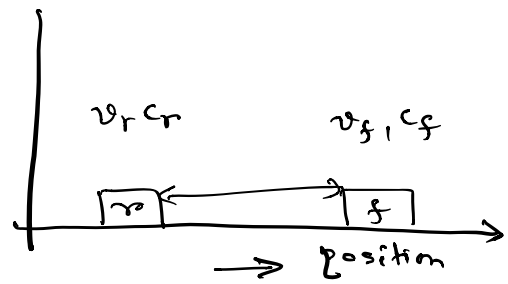
Executions
All behaviors of the model A satisfies the requirement

Find particular ~~behavior~~ Executions of A that violates the Requirement

Example 1 (from last lecture)

Two vehicles on a single lane highway

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




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

$a_{min} < 0 < a_{max}$

$Q = \mathbb{R}^6$

$q \in \mathbb{R}^6 \quad q.v_f \in \mathbb{R}$

$$Q_0 = \begin{cases} c_f[0] = c_r[0] + d_0 \\ v_r[0] = v_0 \end{cases}$$

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

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

refers to $v_f$ component of state $q$

$q'.v_f = q.v_f + a_f.\Delta$ ← $\Delta$ time elapsing

$q'.v_r = q.v_r + a_r.\Delta$

Single transition models $\Delta$ time

$q'.c_f = ---$

$q'.c_r = ---$

Why nondeterminism in the model?

Nondeterminis mechanism for Uncertainty

Sources of Uncertainty:

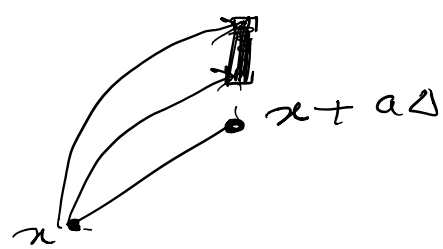
→ Reaction times

→ Sensor noise Range

$x, a$

$$x \xrightarrow{a} x'$$

$$x' = x + \underbrace{a.\Delta} + \varepsilon \quad \text{where } \varepsilon \in [4, 7]$$



State Machine / Discrete transition System / Automaton

$\mathcal{A} = \langle Q, Q_0, A, D \rangle$

$Q$ : Set of States of the automaton

$Q$ is specified by a set of state variables $X$

EX2 $X = \{c_f, v_f, a_f, c_r, v_r, a_r\}$

type $(c_f) = \mathbb{R}$ ... type $(a_f) \in [a_{min}, a_{max}]$

$Q = \mathbb{R}^6$ $= \mathbb{R}$

$Q_0 \subseteq Q$ Set of initial states

$|Q_0|$ not necessarily $= 1$

$A$ : Set of actions or labels

give names to how the states change

$D \subseteq Q \times A \times Q$ $(q, a, q') \in D$ we write

Prestate $q \xrightarrow{a} q'$ Post state

$\exists q_1, q_2$ $q_1 \neq q_2$ s.t $q \xrightarrow{a} q_1 \wedge$

$q \xrightarrow{a} q_2$

$\mathcal{A}$ can be nondeterministic

$\mathcal{A}$ is said to be <u>deterministic</u> if

$\forall q \in Q \ \forall a \ \forall q_1, q_2 \in Q$

if $q \xrightarrow{a} q_1 \wedge q \xrightarrow{a} q_2 \Rightarrow q_1 = q_2$

<u>Post</u>$(q)$ $\qquad$ post : $Q \to 2^Q$

$\triangleq \{ q' \in Q \mid \exists a \in A \ \ q \xrightarrow{a} q' \}$

<u>Post</u>$(Q')$ $= \bigcup_{q \in Q'}$ Post$(q)$

---

## Representation of States.

How to represent states (in programs)

- List $\{1, \ldots 7\}$

$|Q|$ finite $\qquad$ Finite State Machine FSM

$|Q| \sim 2^{100}$

Explicit representation will blow-up

$x_1, x_2 \ldots x_{100} \in \mathbb{B}$

bit vector

Write a formula involving the state variable

$\phi = x_1 \vee x_2 \simeq \{\langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle\}$

$[\![\phi]\!]$ represents the set of states

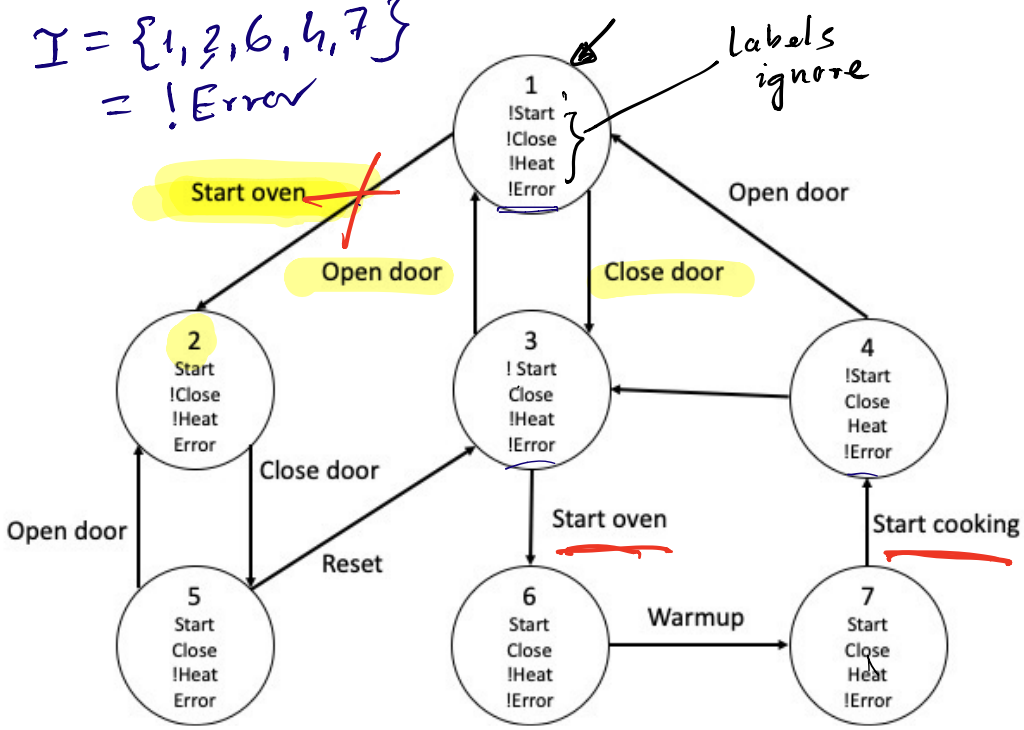
$$Q_\phi = \{ x \mid x \text{ satisfies } \phi \}$$

$$Q_{\phi_1} \cup Q_{\phi_2} \qquad Q_{\phi_2} \cap Q_{\phi_3}$$

$$\phi_1 \vee \phi_2 \qquad \phi_2 \wedge \phi_2$$

# Example 2. Microwave State machine

$I = \{1, 2, 6, 4, 7\}$
$= !Error$




$Q = \{1, 2, \ldots 7\}$

$Q_0 = \{1\}$

$A = \{\text{Start oven}, \text{Reset}, \text{Warmup}, \ldots\}$

$\langle 1, \text{Reset}, 5 \rangle \notin D \quad \langle 1, \text{Reset}, * \rangle \notin D$

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

$D = \{\langle 1, \text{Close door}, 3 \rangle; \langle 1, \text{Start Oven}, 2 \rangle, \ldots\}$

What are Behaviors of an automaton?

$$\mathcal{A} = \langle Q, Q_0, A, D \rangle$$

Executions model a particular run of the automaton $\mathcal{A}$.

An <u>execution</u> of $\mathcal{A}$ is an alternating sequence of states and actions

$$\alpha = q_0 a_1 q_1 a_2 q_2 a_3 \cdots$$

if $\alpha$ is finite $q_0 a_1 q_1 a_2 \cdots q_k$ then $\alpha.\text{lstate} = q_k$

(1) $q_i \in Q \quad \forall i \qquad a_i \in A \quad \forall i$

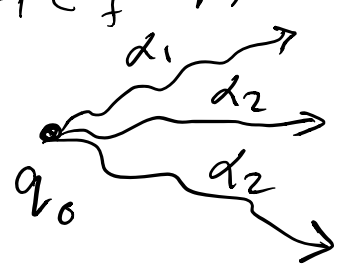
(2) $q_i \xrightarrow{a_{i+1}} q_{i+1}$

(3) $q_0 \in Q_0$

Ex $\alpha = 1$, Start oven, 2, close door, 5, ....

EX1. $\alpha = \underline{x_0} (a_f^1, a_r^1)\, x_1 (a_f^2\, a_r^2) \cdots$

Nondeterministic




Set of all possible executions of the system

Exec. 1.

$\ldots{}_A$

A requirement $R$ is just a subset of $Execs_A$ ; $R \subseteq Execs_A$

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

Reachable states

A state $q \in Q$ is reachable if

$\exists \alpha \in Exec_A \quad \alpha.lstate = q$

$$\boxed{Reach_A = \{q \mid \exists \alpha \in Exec_A \ \ \alpha.lstate = q\}}$$

Requirements

Common type of requirement

Something is always true

OR Something never happens

Invariant

$\boxed{G_f - G_r > d_s}$

Def. An invariant requirement is specified as a set of states $I$ such that $\forall \alpha \in Execs \ \forall i \ \alpha_i \in I$

alternatively $\underline{\text{Reach}_A} \subseteq I$.

<u>Ex1</u> if we choose $c_f[o] - c_r[o] > d_o$

and $\underline{\forall t}$ $v_f[t], v_r[t] > 0$

then

$\forall t$ $\boxed{c_f[t] - c_r[t] > 0}$

invariant.

```
Set Marked := {}
Queue Q := S
Marked := Marked ∪ S
while Q is not empty
        t ← Q.dequeue()
        if t ∈ T return "yes"
        for each (t,u) ∈ E
                if u ∉ Marked then
                Marked := Marked ∪ {u}
                Q := enqueue(Q, u)
return "no"
```