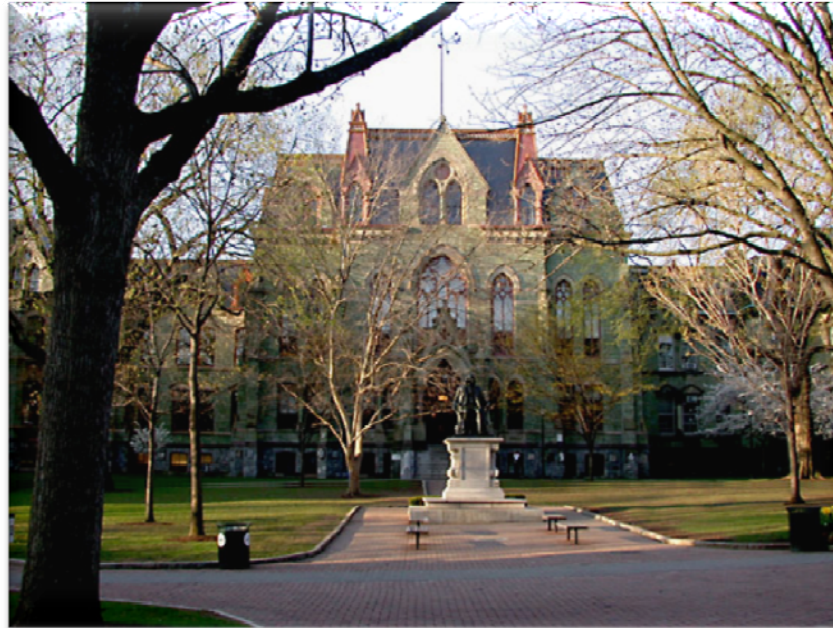


ESE601 - Hybrid Systems Discrete Systems Review



George J. Pappas

Department of Electrical and Systems Engineering

University of Pennsylvania

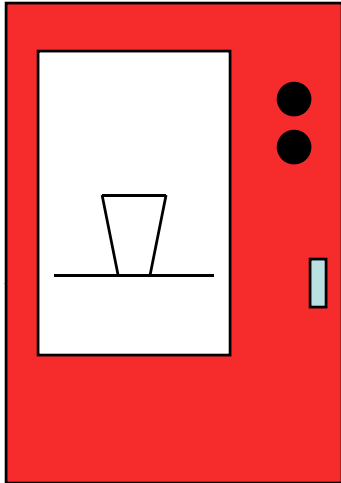
pappasg@seas.upenn.edu



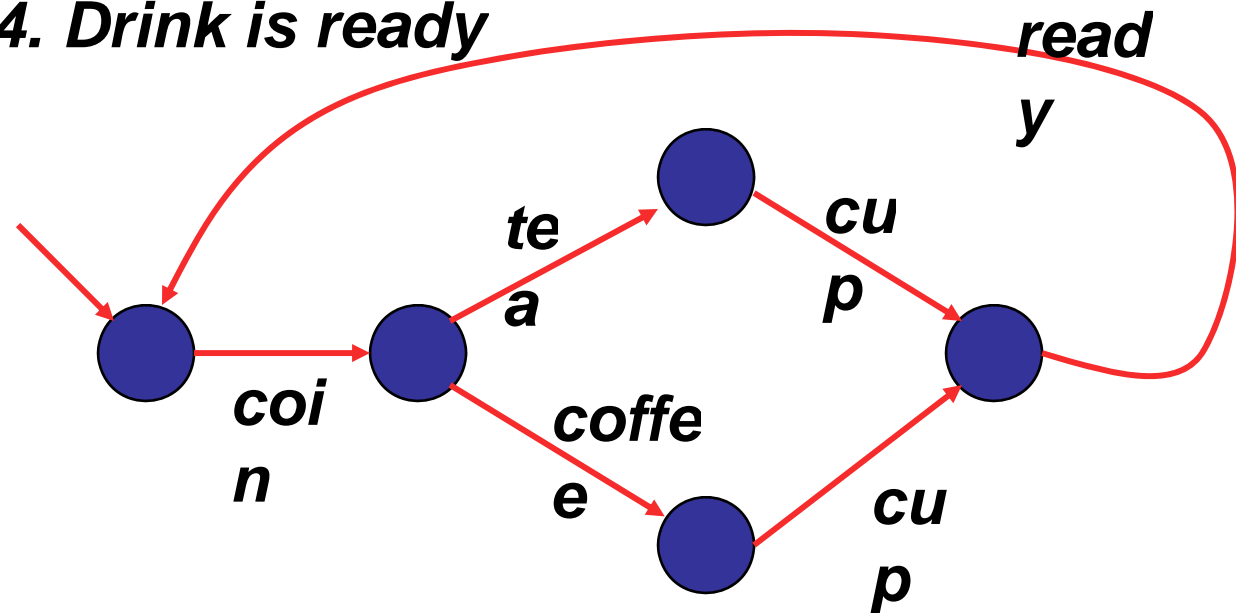
Contents

- Examples (5)
- Transition systems and automata (6)
- Languages and regular expression (6)
- Reachability and blocking property (5)
- Composition of automata (5)
- I/O automata (5)

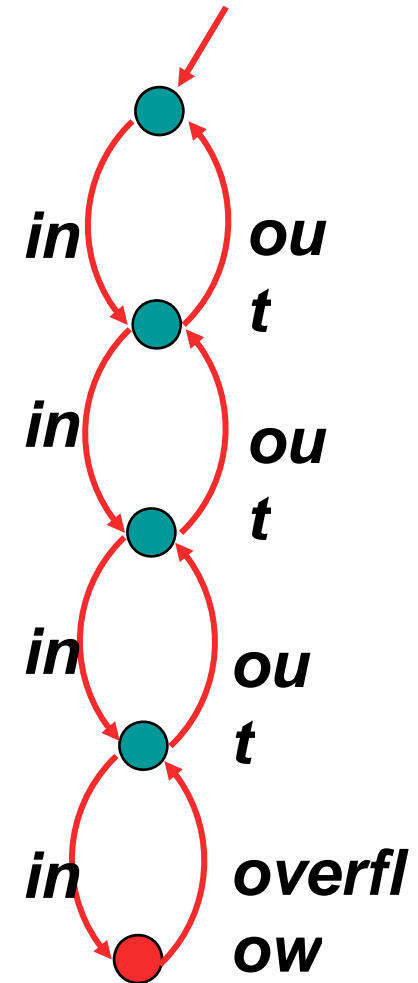
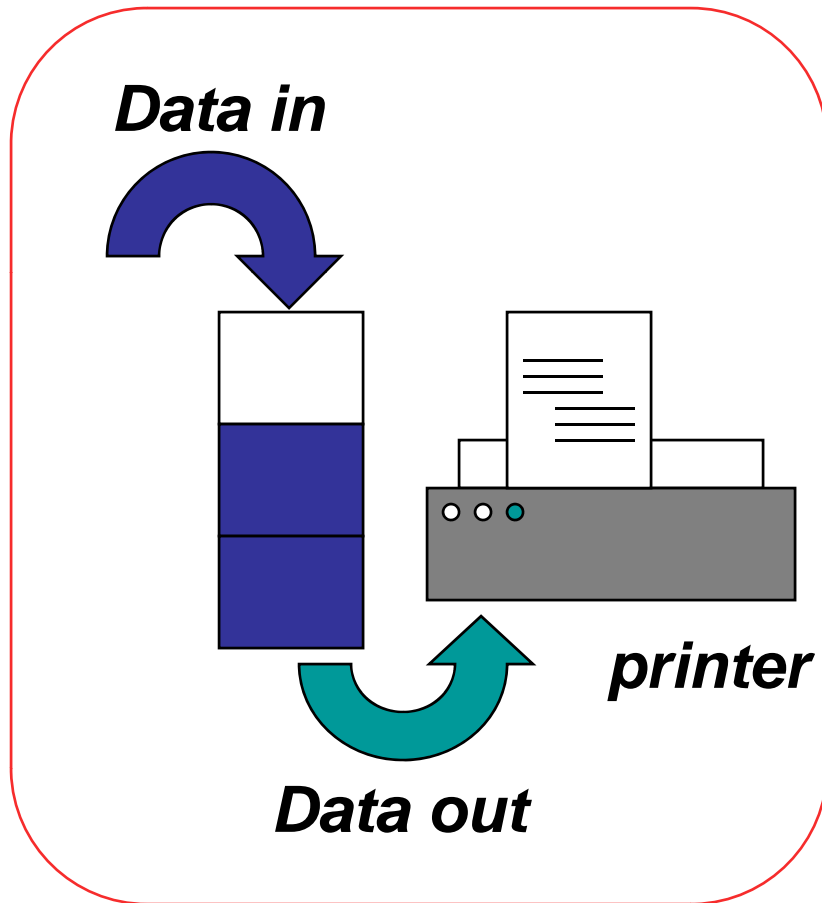
A vending machine



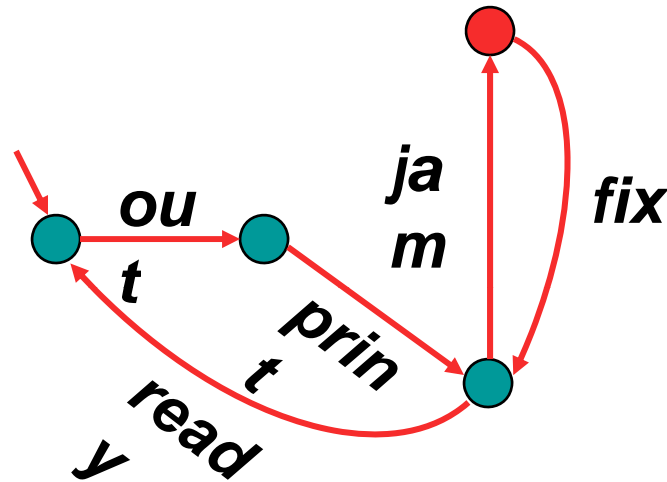
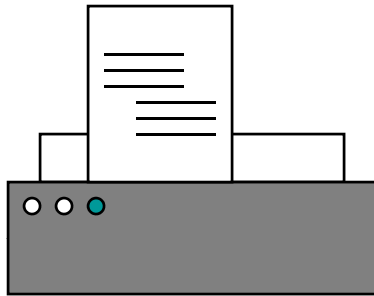
- 1. Insert coin(s)**
- 2. Choose tea or coffee**
- 3. Put the cup on the tray**
- 4. Drink is ready**



Printer data buffer

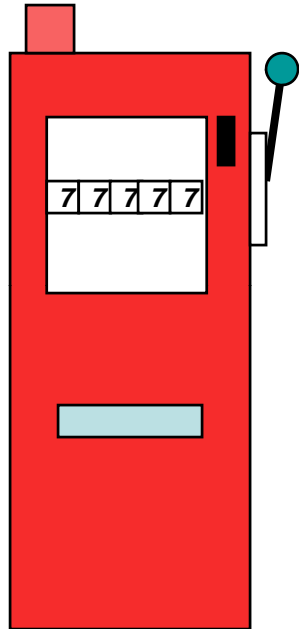


A printer

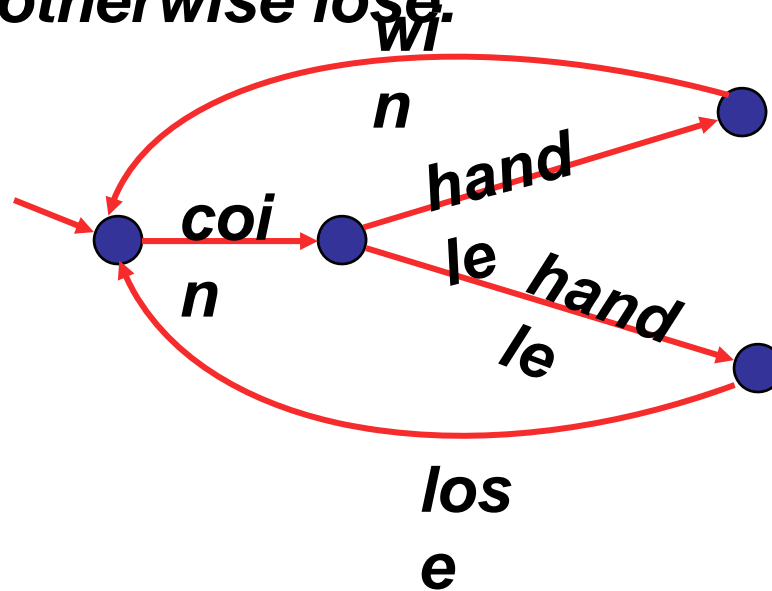


The printer receives data from the buffer, and print it out. Once the printout is ready, the printer is ready to receive new data. While printing, the paper can jam and need to be fixed before the printing process can resume.

A slot machine



1. *Insert coin*
2. *Pull handle*
3. *Win if the combination is good, otherwise lose.*



Modeling recap

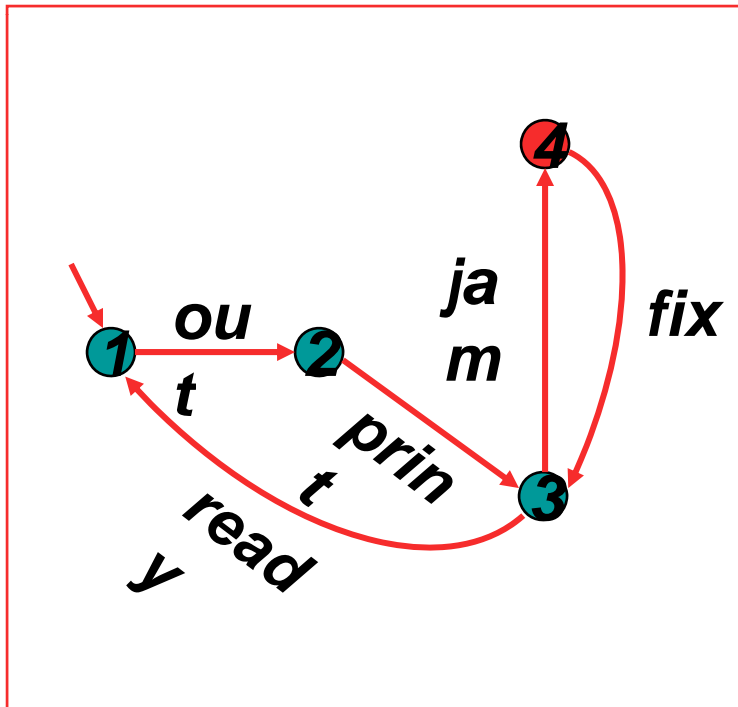
- Events are **time-abstract**.
- Just like modeling of continuous systems, the level of detail is '**modeler dependent**'.
- Events are not necessarily equipped with any notion of 'internal-external' or 'input-output'.
- **Compositionality** is possible (to be discussed later).
- There can be **non-determinism**.

Transition systems

- A transition system $S = (Q, A, \rightarrow, Y, \langle \cdot \rangle, Q_0)$, where:
 - Q is the set of **states**,
 - A is the set of **labels**,
 - $\rightarrow \subset Q \times A \times Q$ is the set of **transitions**,
 - Y is the set of **observations/outputs**,
 - $\langle \cdot \rangle : Q \rightarrow Y$ is the **observation/output map**,
 - Q_0 is the set of initial **states**.
- When there is a transition from the state $q \in Q$ to $q' \in Q$ with label $a \in A$, we denote it by $q \xrightarrow{a} q'$.
- When Q and A are finite, we called the transition system **finite**.

Transition systems

- If for some $q \in Q$ and $a \in A$ there can be more than one $q' \in Q$ such that $q \xrightarrow{a} q'$, then the transition system is **nondeterministic**.



$$Q = \{1, 2, 3, 4\}$$

$$Q_0 = 1$$

$$A = \{out, print, ready, jam, fix\}$$

$$Y = \{normal, error\}$$

$$\langle 1 \rangle = \langle 2 \rangle = \langle 3 \rangle = normal$$

$$\langle 4 \rangle = error$$

Execution of transition systems

- Given a transition system $S = (Q, A, \rightarrow, Y, \langle \cdot \rangle, Q_0)$, a sequence

$$q_0 a_0 q_1 a_1 \cdots q_{N+1},$$
$$(q_i)_{1 \leq i \leq N+1} \in Q, (a_i)_{1 \leq i \leq N} \in A,$$

is an **execution** of S , if

$$q_0 \in Q_0 \text{ and } q_i \xrightarrow{a_i} q_{i+1}, 1 \leq i \leq N.$$

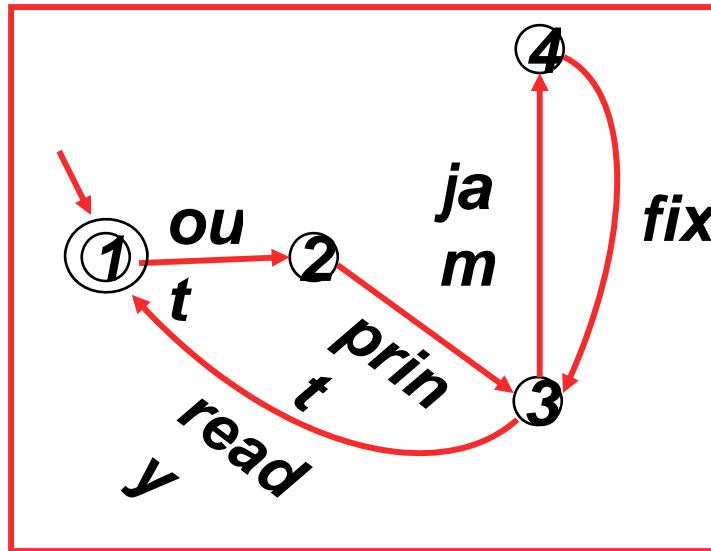
- Given an execution $q_0 a_0 q_1 a_1 \cdots q_{N+1}$, the **external trajectory** associated with it is given by $\langle q_0 \rangle a_0 \langle q_1 \rangle a_1 \cdots \langle q_{N+1} \rangle$.
- The collection of all external trajectories of the transition system is called the **language** of the system.

Automata

- An automaton $\mathcal{A} = (Q, A, \rightarrow, Q_0, Q_m)$, where:
 - Q is the set of **states**,
 - A is the set of **labels**,
 - $\rightarrow \subset Q \times A \times Q$ is the set of **transitions**,
 - $Q_0 \subset Q$ is the set of **initial states**,
 - $Q_m \subset Q$ is the set of **marked states**.
- When the set Q is finite, the automaton is a **finite state automaton**.
- Marked states are states that have a special property, typically corresponding to a proper termination of a function.

Automata

- When there is a transition from the state $q \in Q$ to $q' \in Q$ with label $a \in A$, we denote it by $q \xrightarrow{a} q'$.
- If for some $q \in Q$ and $a \in A$ there can be more than one $q' \in Q$ such that $q \xrightarrow{a} q'$, then the automaton is **nondeterministic**.



$$Q = \{1, 2, 3, 4\}$$

$$Q_0 = \{1\}$$

$$A = \{out, print, ready, jam, fix\}$$

$$Q_m = \{1\}$$

Execution of automata

- Given an automaton $\mathfrak{A} = (Q, A, \rightarrow, Q_0, Q_m)$, a sequence

$$q_0 a_0 q_1 a_1 \cdots q_{N+1},$$
$$(q_i)_{1 \leq i \leq N+1} \in Q, (a_i)_{1 \leq i \leq N} \in A,$$

is an **execution** of \mathfrak{A} , if

$$q_0 \in Q_0 \text{ and } q_i \xrightarrow{a_i} q_{i+1}, 1 \leq i \leq N.$$

- Given an execution $q_0 a_0 q_1 a_1 \cdots q_{N+1}$, the **trace** associated with it is given by $a_0 a_1 a_2 a_3 \cdots a_N$.
- The collection of all traces of the automaton is called the **generated language** of the automaton, $L(\mathfrak{A})$.

Regular languages

- An **alphabet** A is a finite collection of symbols.
- A **string** s over the alphabet A is a sequence of symbols in A . The empty string is denoted by ϵ .
- The **length of a string** s is denoted by $|s|$. The length of the empty string is zero.
- A **language** L over the alphabet A is a collection of finite-length strings over A .
- Strings over an alphabet can be concatenated to form a longer string. Eg: $s_1 = aba, s_2 = bba$, then $s_1s_2 = ababba$. The empty string is the identity element of the concatenation.

Regular languages

- Other than the usual set operations, there are other basic operations on languages.
- Let L_a and L_b be languages over the alphabet A , then the **concatenation**

$$L_a L_b := \{s \mid \exists s_a \in L_a, s_b \in L_b, s = s_a s_b\}.$$

- Let L be a language over the alphabet A , then the **prefix closure** of L is defined as

$$\bar{L} := \{s \mid \exists s' \text{ s.t. } ss' \in L\}.$$

- Let L be a language over the alphabet A , then the **Kleene closure** of L is defined as

$$L^* := \{\varepsilon\} \cup L \cup LL \cup \dots$$

Regular languages

- When a language L is such that $L = \bar{L}$, it is called **prefix closed**.
- The **language generated** by a finite state automaton is always prefix closed.
- An execution of the automaton \mathcal{A} is a **marked execution** if it terminates at a marked state.
- A trace corresponding to a marked execution is called a **marked trace**.
- The collection of the marked trace of an automaton \mathcal{A} is called the **accepted language** of the automaton, $L_m(\mathcal{A})$.

Regular languages

The set of languages accepted by finite state automata is called the **regular languages**.

THEOREM: Given a regular language L , there exists an integer n such that any string $s \in L$, with $|s| \geq n$, can be broken into three strings $s = xyz$, where

- (i) $y \neq \epsilon$,
- (ii) $|xy| \leq n$,
- (iii) For all integers $k > 0$, the string $xy^kz \in L$.

Regular expressions

- A regular expressions can be thought of as a formula corresponding to a regular language. If p is a regular expression, then the regular language corresponding to p is denoted as $L(p)$.
- For any symbol $a \in A$, the regular expression a corresponds to the language $\{a\}$. The expression ε corresponds to the language $\{\varepsilon\}$. The expression \emptyset corresponds to the empty language $\{\}$.

$$L(p_1 + p_2) = L(p_1) \cup L(p_2),$$

$$L(p_1 \cdot p_2) = L(p_1)L(p_2),$$

$$L(p^*) = (L(p))^*,$$

$$L(\bar{p}) = \overline{L(p)}.$$

Accessibility

- Given an automaton $\mathcal{A} = (Q, A, \rightarrow, Q_0, Q_m)$, a state $q \in Q$ is **accessible** if there is an execution that reaches q .
- Removing the states that are not accessible **does not affect** the language generated and accepted by the automaton.
- The automaton obtained by removing the non-accessible states of \mathcal{A} is denoted by $Ac(\mathcal{A})$.
- If $\mathcal{A} = Ac(\mathcal{A})$, then \mathcal{A} is **accessible**.

Blocking property

- Given an automaton $\mathfrak{A} = (Q, A, \rightarrow, Q_0, Q_m)$, the automaton is **blocking** if

$$\overline{L_m(\mathfrak{A})} \neq L(\mathfrak{A})$$

- Notice that in general

$$L_m(\mathfrak{A}) \subseteq \overline{L_m(\mathfrak{A})} \subseteq L(\mathfrak{A})$$

- Intpretation: Blocking means there is an execution that cannot be continued to reach a marked state.

Co-accessibility

- Given an automaton $\mathfrak{A} = (Q, A, \rightarrow, Q_0, Q_m)$, a state $q \in Q$ is **co-accessible** if there is an execution that starts from q and reach a marked state.
- The automaton obtained by removing states that are not co-accessible in \mathfrak{A} is denoted by $CoAc(\mathfrak{A})$. If $\mathfrak{A} = CoAc(\mathfrak{A})$, we say that \mathfrak{A} is co-accessible.
- We have that

$$\begin{aligned}L(CoAc(\mathfrak{A})) &\subset L(\mathfrak{A}), \\L_m(CoAc(\mathfrak{A})) &= L_m(\mathfrak{A}).\end{aligned}$$

- An automaton is blocking if and only if it is not co-accessible.

Co-accessibility

- Blocking is a property of the **languages**, while co-accessibility is a property of the **automaton**.
- Blocking means any automaton realization of the languages is not co-accessible.
- The automaton $Ac(CoAc(\mathcal{A})) = CoAc(Ac(\mathcal{A}))$ is called the trimmed version of \mathcal{A} , $Trim(\mathcal{A})$. If $\mathcal{A} = Trim(\mathcal{A})$, then \mathcal{A} is **trim**.

Composition of automata

- Given two automata $\mathfrak{A}_i = (Q_i, A, \rightarrow_i, Q_{0,i}, Q_{m,i})$, $i = 1, 2$. The composed automaton

$$\mathfrak{A} = \mathfrak{A}_1 \parallel \mathfrak{A}_2,$$

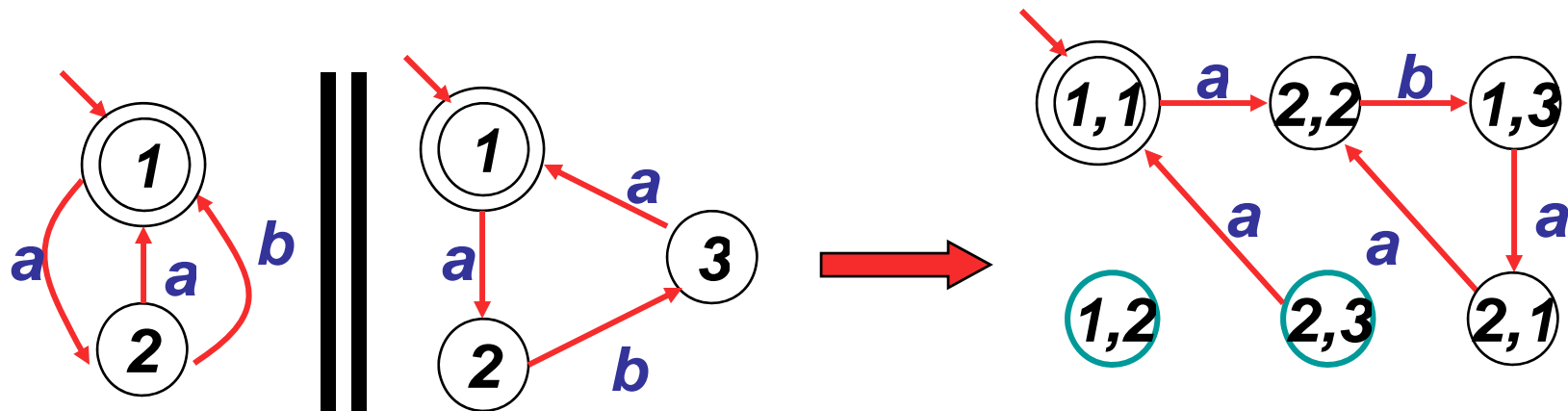
where $\mathfrak{A} = (Q_1 \times Q_2, A, \rightarrow, Q_{0,1} \times Q_{0,2}, Q_{m,1} \times Q_{m,2})$,
where

$$(q_1, q_2) \xrightarrow{a} (q'_1, q'_2) \text{ iff } q_i \xrightarrow{a}_i q'_i, i = 1, 2.$$

- **Notice: Same set of labels.**
- The set of states of the composed automaton is the **product** of those of the components.

Composition of automata

- An initial state of the composed automaton is a **product** of initial states of the components.
- A state is marked iff it is marked in both components.
- Interpretation: Both components have to **synchronize** on each event, while performing their tasks.



Composition of automata

- The relations between composition and the languages of the automata:

$$L(\mathcal{A}_1 \parallel \mathcal{A}_2) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2),$$
$$L_m(\mathcal{A}_1 \parallel \mathcal{A}_2) = L_m(\mathcal{A}_1) \cap L_m(\mathcal{A}_2).$$

- Non-blockingness is not preserved.

Composition of automata

Given two automata $\mathfrak{A}_i = (Q_i, A_i, \rightarrow_i, Q_{0,i}, Q_{m,i})$, $i = 1, 2$,
where

$$A_1 \cap A_2 =: A.$$

The composed automaton

$$\mathfrak{A} = \mathfrak{A}_1 \parallel \mathfrak{A}_2,$$

where $\mathfrak{A} = (Q_1 \times Q_2, A_1 \cup A_2, \rightarrow, Q_{0,1} \times Q_{0,2}, Q_{m,1} \times Q_{m,2})$,
where

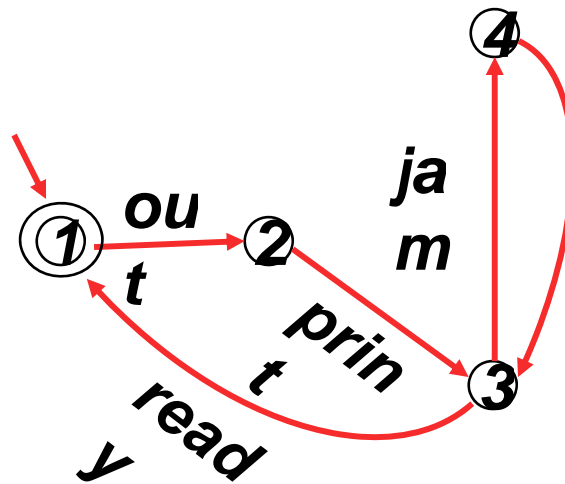
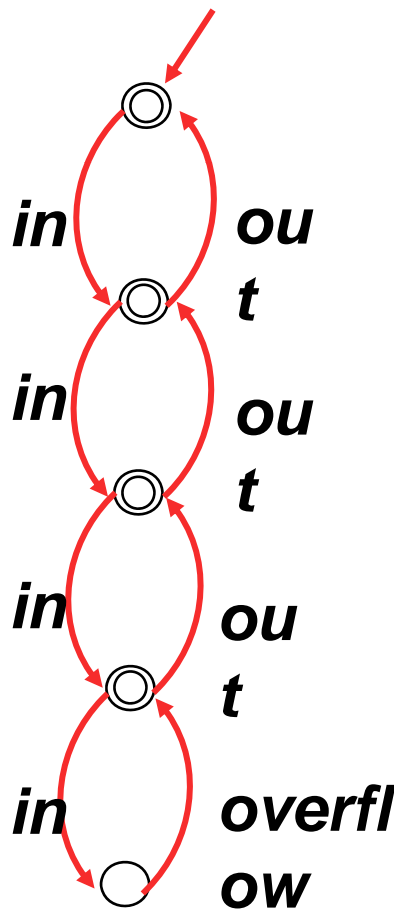
$$(q_1, q_2) \xrightarrow{a} (q'_1, q'_2) \text{ iff } a \in A, q_i \xrightarrow{a}_i q'_i, i = 1, 2,$$

$$(q_1, q_2) \xrightarrow{a} (q'_1, q_2) \text{ iff } a \notin A, q_1 \xrightarrow{a}_1 q'_1,$$

$$(q_1, q_2) \xrightarrow{a} (q_1, q'_2) \text{ iff } a \notin A, q_2 \xrightarrow{a}_2 q'_2.$$

Composition of automata

- Thus, the automata synchronize only on the events that they have in common.



Composition and languages

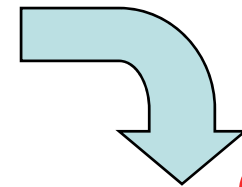
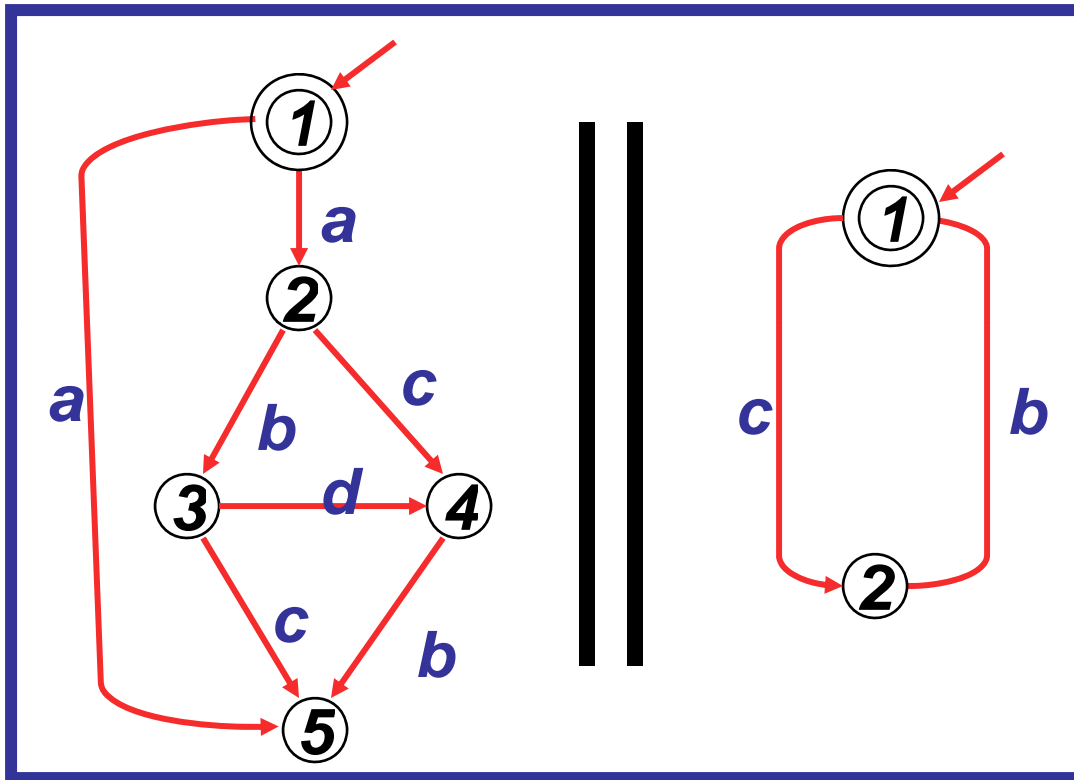
- Let L be a regular language over the alphabet A . Let E be a subset of A . We define the **projection** $\pi_E : L \rightarrow E^*$, such that $\pi_E(s)$ returns the string created by **removing all symbols in s that are not elements of E** .
- Example: $A = \{a, b, c\}, E = \{a, b\}$,

$$\pi_E(abcabcccab) = ababab,$$

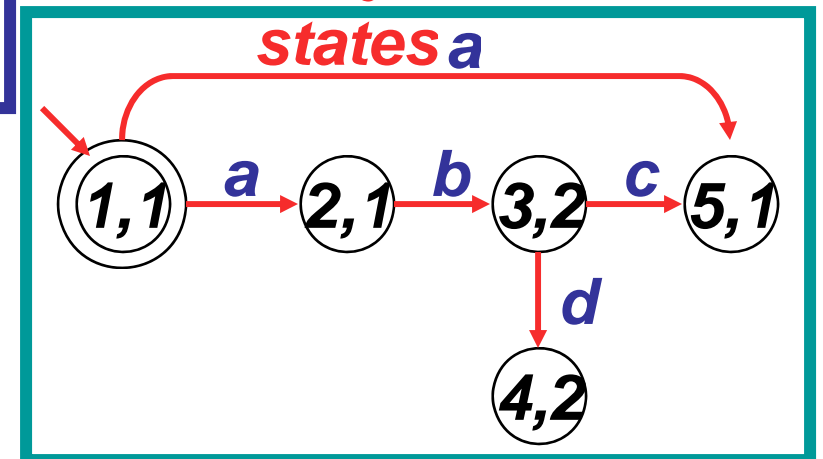
$$\pi_E(accc) = a,$$

$$\pi_E(ccccc) = \varepsilon.$$

Example



Only accessible states a



Composition and languages

The languages of the composed automaton: For any $s \in (A_1 \cup A_2)^*$,

$$s \in L(\mathfrak{A}_1 \parallel \mathfrak{A}_2) \Leftrightarrow \begin{cases} \pi_{A_i}(s) \in L(\mathfrak{A}_i), i = 1, 2, \\ \pi_A(s) \in \pi_A(L(\mathfrak{A}_1)) \cap \pi_A(L(\mathfrak{A}_2)). \end{cases}$$

$$s \in L_m(\mathfrak{A}_1 \parallel \mathfrak{A}_2) \Leftrightarrow \begin{cases} \pi_{A_i}(s) \in L_m(\mathfrak{A}_i), i = 1, 2, \\ \pi_A(s) \in \pi_A(L_m(\mathfrak{A}_1)) \cap \pi_A(L_m(\mathfrak{A}_2)). \end{cases}$$

Input – output automata

- So far we have seen composition of automata in the **synchronization style**, i.e. there is no sense of input or output, or which component initiates the event, etc.
- Sometimes it is desirable to model explicitly the sense of input and output in the composition.
- Input - output automata: the set of events is partitioned into inputs $in(\mathcal{A})$, outputs $out(\mathcal{A})$, and internal events $int(\mathcal{A})$.

Input – output automata

- Every state is **input enabled**. For each input event, there exists an outgoing transition with that label.
- A composition of two input-output automata \mathcal{A}_1 and \mathcal{A}_2 is **compatible** if

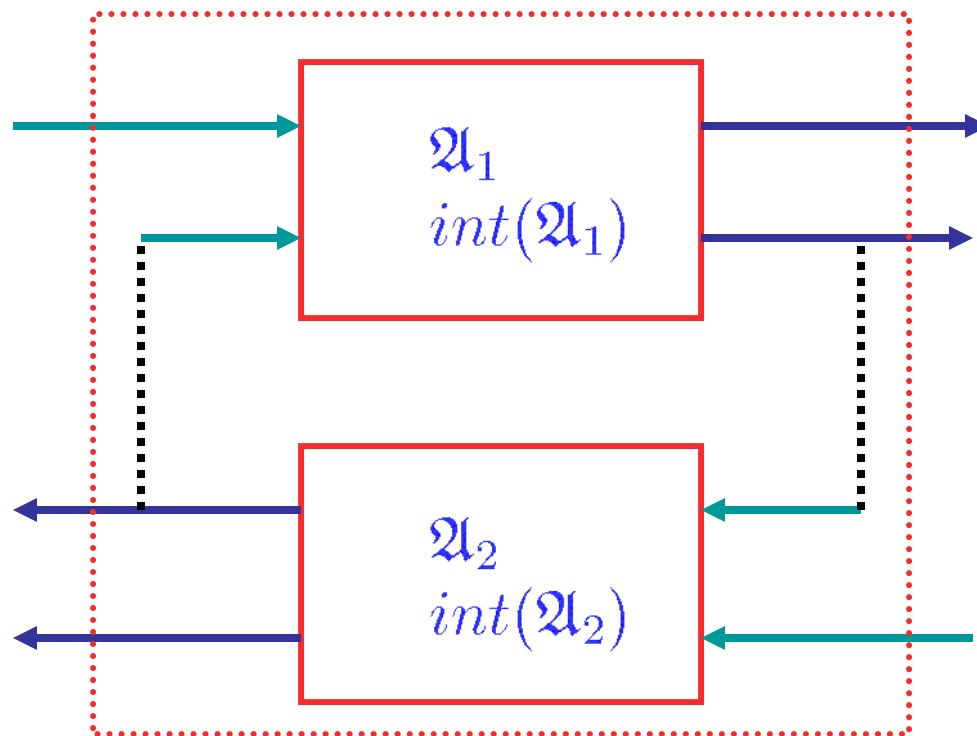
$$\begin{aligned} out(\mathcal{A}_1) \cap out(\mathcal{A}_2) &= \emptyset \\ int(\mathcal{A}_1) \cap (in(\mathcal{A}_2) \cup out(\mathcal{A}_2) \cup int(\mathcal{A}_2)) &= \emptyset \\ int(\mathcal{A}_2) \cap (in(\mathcal{A}_1) \cup out(\mathcal{A}_1) \cup int(\mathcal{A}_1)) &= \emptyset \end{aligned}$$

- If that is the case then,

$$\begin{aligned} out(\mathcal{A}_1 \parallel \mathcal{A}_2) &= out(\mathcal{A}_1) \cup out(\mathcal{A}_2) \\ int(\mathcal{A}_1 \parallel \mathcal{A}_2) &= int(\mathcal{A}_1) \cup int(\mathcal{A}_2) \end{aligned}$$

Input – output automata

Compatibility implies some kind of **feedback structure**.



Transition Systems

A transition system

$$T = (Q, \Sigma, \rightarrow, O, \langle \cdot \rangle)$$

consists of

A set of states Q

A set of events Σ

A set of observations O

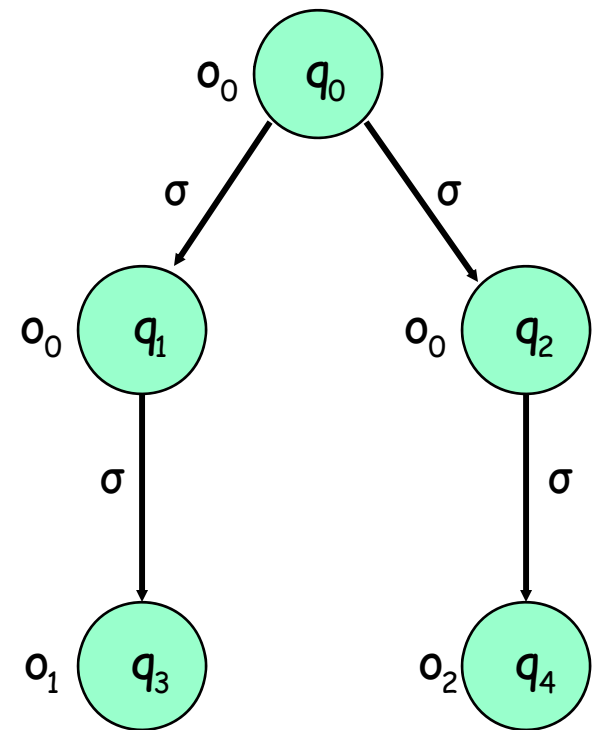
The transition relation $q_1 \xrightarrow{\sigma} q_2$

The observation map $\langle q_1 \rangle = o_0$

Initial or final states may be incorporated

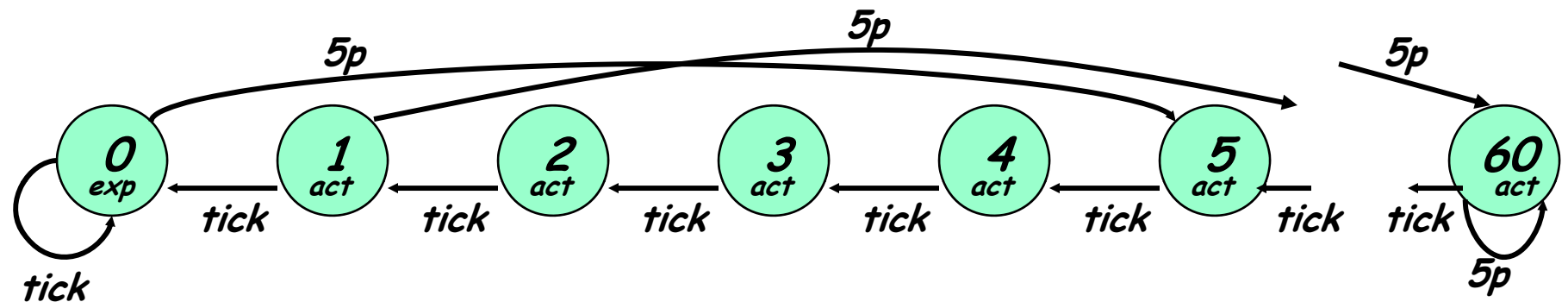
The sets Q , Σ , and O may be infinite

Language of T is all sequences of observations



A painful example

The parking meter



States $Q = \{0, 1, 2, \dots, 60\}$

Events $\{\text{tick}, 5p\}$

Observations $\{\text{exp}, \text{act}\}$

A possible string of observations $(\text{exp}, \text{act}, \text{act}, \text{act}, \text{act}, \text{act}, \text{exp}, \dots)$

A familiar example

$$T^\Delta = (Q, \Sigma, \rightarrow, O, \langle \cdot \rangle)$$

 Δ

$$x_{k+1} = Ax_k + Bu_k$$

$$y_k = Cx_k$$

Transition System T^Δ

State set $Q = X = \mathbb{R}^n$

Label set $\Sigma = U = \mathbb{R}^m$

Observation set $O = Y = \mathbb{R}^p$

Linear Observation Map $\langle x \rangle = Cx$

Transition Relation $\rightarrow \subseteq X \times U \times X$

$$x_1 \xrightarrow{u} x_2 \Leftrightarrow x_2 = Ax_1 + Bu$$

Loose Control...

$$T_{\Sigma}^{\Delta} = (Q, \Sigma, \rightarrow, O, \langle \cdot \rangle)$$

$$\begin{aligned} \Delta \quad x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned}$$

Transition System T_1^{Δ}
State set $Q = X = \mathbb{R}^n$
Label set $\Sigma = \{1\}$
Observation set $O = Y = \mathbb{R}^p$
Linear Observation Map $\langle x \rangle = Cx$
Transition Relation $\rightarrow \subseteq X \times \{1\} \times X$
$x_1 \xrightarrow{1} x_2 \Leftrightarrow \exists u \text{ with } x_2 = Ax_1 + Bu$

Keep time....

$$T_{\Sigma}^{\Delta} = (Q, \Sigma, \rightarrow, O, \langle \cdot \rangle)$$

$$\begin{aligned} \Delta \quad x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned}$$

Transition System $T_{N_+}^{\Delta}$

State set $Q = X = \mathbb{R}^n$

Label set $\Sigma = N_+$

Observation set $O = Y = \mathbb{R}^p$

Linear Observation Map $\langle x \rangle = Cx$

Transition Relation $\rightarrow \subseteq X \times N_+ \times X$

$$x_1 \xrightarrow{k} x_2 \Leftrightarrow \exists u_0, \dots, u_{k-1} \text{ with}$$
$$x_2 = A^k x_1 + \sum_{i=0}^{k-1} A^{k-i-1} B u_i$$

Loose control and time...

$$T_{\Sigma}^{\Delta} = (Q, \Sigma, \rightarrow, O, \langle \cdot \rangle)$$

$$\begin{aligned} \Delta \quad x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned}$$

Transition System T_{τ}^{Δ}

State set $Q = X = \mathbb{R}^n$

Label set $\Sigma = \{\tau\}$

Observation set $O = Y = \mathbb{R}^p$

Linear Observation Map $\langle x \rangle = Cx$

Transition Relation $\rightarrow \subseteq X \times \{\tau\} \times X$

$x_1 \xrightarrow{\tau} x_2 \Leftrightarrow \exists k \text{ and } \exists u_0, \dots, u_{k-1} \text{ with}$

$$x_2 = A^k x_1 + \sum_{i=0}^{k-1} A^{k-i-1} B u_i$$

Finite Observations

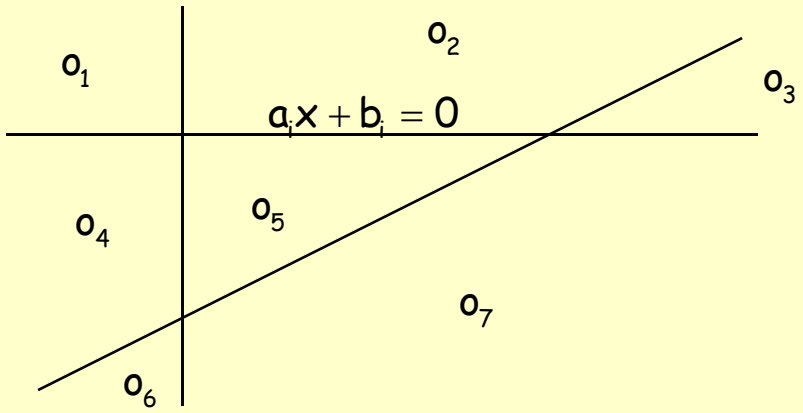
$$T_{\Sigma}^{\Delta} = (Q, \Sigma, \rightarrow, O, \langle \cdot \rangle)$$

$$\Delta \quad \begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned}$$

All Transition Systems

Finite Observations $O = \{o_1, o_2, \dots, o_p\}$

Polyhedral Map $\langle x \rangle : X \rightarrow O$



Keep continuous time....

$$T_{\Sigma}^{\Delta} = (Q, \Sigma, \rightarrow, O, \langle \cdot \rangle)$$

$$\begin{aligned} \Delta \\ x' &= Ax + Bu \\ y &= Cx \end{aligned}$$

$$\begin{aligned} &\text{Transition System } T_{R_+}^{\Delta} \\ &\text{State set } Q = X = R^n \\ &\text{Label set } \Sigma = R_+ \\ &\text{Observation set } O = Y = R^p \\ &\text{Linear Observation Map } \langle x \rangle = Cx \\ &\text{Transition Relation } \rightarrow \subseteq X \times R_+ \times X \\ &\quad \exists u_{[0,t]} \text{ with} \\ x_1 \xrightarrow{+} x_2 &\Leftrightarrow x_2 = e^{At}x_1 + \int_0^t e^{A(t-s)}Bu(s)ds \end{aligned}$$

Loose continuous time....

$$T_{\Sigma}^{\Delta} = (Q, \Sigma, \rightarrow, O, \langle \cdot \rangle)$$

$$\begin{array}{l} \Delta \\ x' = Ax + Bu \\ y = Cx \end{array}$$

Transition System T_{τ}^{Δ}
State set $Q = X = \mathbb{R}^n$
Label set $\Sigma = \{\tau\}$
Observation set $O = Y = \mathbb{R}^p$
Linear Observation Map $\langle x \rangle = Cx$
Transition Relation $\rightarrow \subseteq X \times \{\tau\} \times X$
$x_1 \xrightarrow{\tau} x_2 \Leftrightarrow \exists t \text{ and } \exists u_{[0,t]} \text{ with}$
$x_2 = e^{At}x_1 + \int_0^t e^{A(t-s)}Bu(s)ds$

Transition Systems

A region is a subset of states $P \subseteq Q$

We define the following operators

$$\text{Pre}_\sigma(P) = \{q \in Q \mid \exists p \in P \quad q \xrightarrow{\sigma} p\}$$

$$\text{Pre}(P) = \{q \in Q \mid \exists \sigma \in \Sigma \quad \exists p \in P \quad q \xrightarrow{\sigma} p\}$$

$$\text{Post}_\sigma(P) = \{q \in Q \mid \exists p \in P \quad p \xrightarrow{\sigma} q\}$$

$$\text{Post}(P) = \{q \in Q \mid \exists \sigma \in \Sigma \quad \exists p \in P \quad p \xrightarrow{\sigma} q\}$$

Transition Systems

We can recursively define

$$\text{Pre}_\sigma^1(P) = \text{Pre}_\sigma(P)$$

$$\text{Pre}_\sigma^n(P) = \text{Pre}_\sigma(\text{Pre}_\sigma^{n-1}(P))$$

Similarly for the other operators. Also

$$\text{Pre}^*(P) = \bigcup_{n \in \mathbb{N}} \text{Pre}^n(P)$$

$$\text{Post}^*(P) = \bigcup_{n \in \mathbb{N}} \text{Post}^n(P)$$

Basic safety problems

Given transition system T and regions P, S determine

Forward Reachability

$$Post^*(P) \cap S \neq \emptyset$$

Backward Reachability

$$P \cap Pre^*(S) \neq \emptyset$$

Forward reachability algorithm

Forward Reachability Algorithm

```
initialize  $R := P$   
while TRUE do  
  if  $R \cap S \neq \emptyset$  return UNSAFE ; end if;  
  if  $Post(R) \subseteq R$  return SAFE ; end if;  
   $R := R \cup Post(R)$   
end while
```

If T is finite, then algorithm terminates (decidability).

Complexity : $O(n_I + m_R)$



↑
initial
states

↑
reachable
transitions

Backward reachability algorithm

Backward Reachability Algorithm

```
initialize  $R := S$   
while TRUE do  
  if  $R \cap P \neq \emptyset$  return UNSAFE ; end if;  
  if  $Pre(R) \subseteq R$  return SAFE ; end if;  
   $R := R \cup Pre(R)$   
end while
```

If T is infinite, then there is no guarantee of termination.

Algorithmic issues

Representation issues

Enumeration for finite sets

Symbolic representation for infinite (or finite) sets

Operations on sets

Boolean operations

Pre and Post computations (closure?)

Algorithmic termination (decidability)

Guaranteed for finite transition systems

No guarantee for infinite transition systems