Discrete Synthesis

Paulo Tabuada

Cyber-Physical Systems Laboratory
Department of Electrical Engineering
University of California at Los Angeles

CyPhyLab UCLA

Systems

Definitions

Formal models for software typically have finitely many states. Think of a finite-state machine model for a protocol.

Systems

Definitions

Formal models for software typically have finitely many states. Think of a finite-state machine model for a protocol.

Formal models for control typically have infinitely many states. Think of a differential equation model for an electric circuit or a mechanical system.

Systems

Definitions

Formal models for software typically have finitely many states. Think of a finite-state machine model for a protocol.

Formal models for control typically have infinitely many states. Think of a differential equation model for an electric circuit or a mechanical system.

In this workshop we will use the notion of *system* to capture both finite-state as well as infinite-state models of dynamic phenomena.

CyPhyLab UCLA

Systems

Definitions

Formal models for software typically have finitely many states. Think of a finite-state machine model for a protocol.

Formal models for control typically have infinitely many states. Think of a differential equation model for an electric circuit or a mechanical system.

In this workshop we will use the notion of *system* to capture both finite-state as well as infinite-state models of dynamic phenomena.

Definition (System)

A *system* S is a sextuple $(X, X_0, U, \longrightarrow, Y, H)$ consisting of:

- a set of *states* X;
- a set of *initial states* $X_0 \subseteq X$;
- a set of *inputs* U;
- a *transition relation* $\longrightarrow \subseteq X \times U \times X$;
- a set of *outputs* Y;
- an *output map* $H : X \to Y$.

Definition (System)

A *system* S is a sextuple $(X, X_0, U, \longrightarrow, Y, H)$ consisting of:

- a set of *states* X;
- a set of *initial states* $X_0 \subseteq X$;
- a set of *inputs* U;
- a *transition relation* $\longrightarrow \subseteq X \times U \times X$;
- a set of *outputs* Y;
- an *output map* $H : X \to Y$.

A system is called *finite-state* if X is a finite set. A system that is not finite-state is called *infinite-state*.

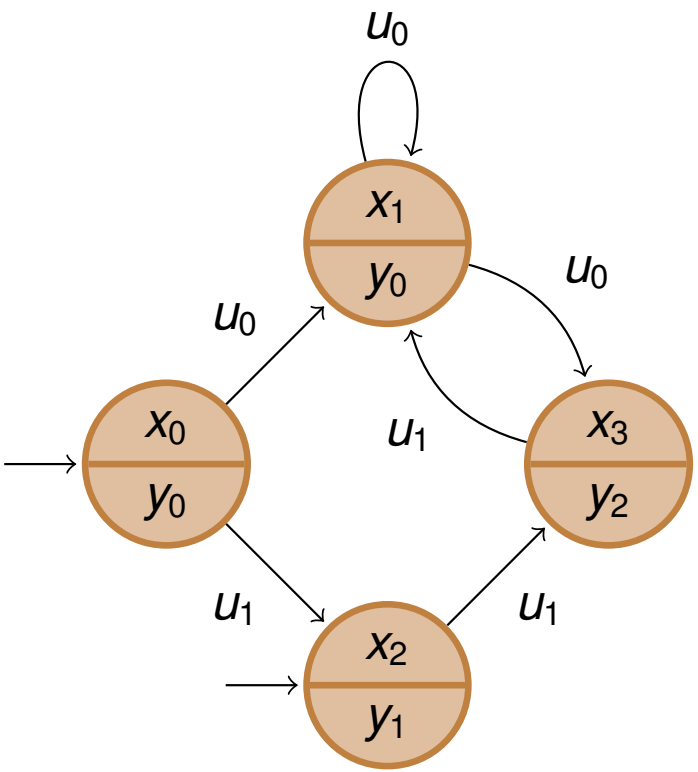
Definition (System)

A *system* S is a sextuple $(X, X_0, U, \longrightarrow, Y, H)$ consisting of:

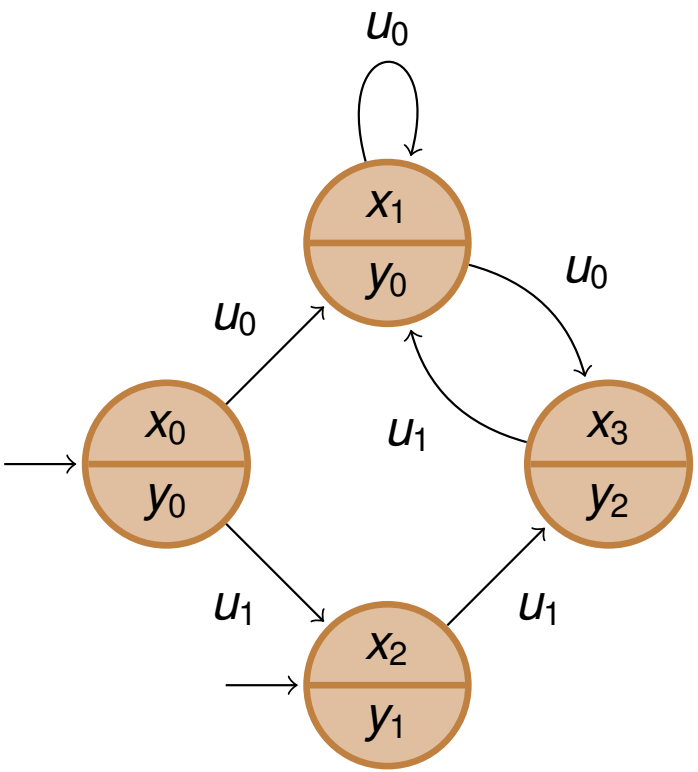
- a set of *states* X;
- a set of *initial states* $X_0 \subseteq X$;
- a set of *inputs* U;
- a *transition relation* $\longrightarrow \subseteq X \times U \times X$;
- a set of *outputs* Y;
- an *output map* $H : X \to Y$.

The evolution of a system is captured by the transition relation. A transition $(x, u, x') \in \longrightarrow$ is denoted by $x \xrightarrow{u} x'$. We can also regard the transition relation as a set-valued map $\delta : X \times U \to 2^X$.

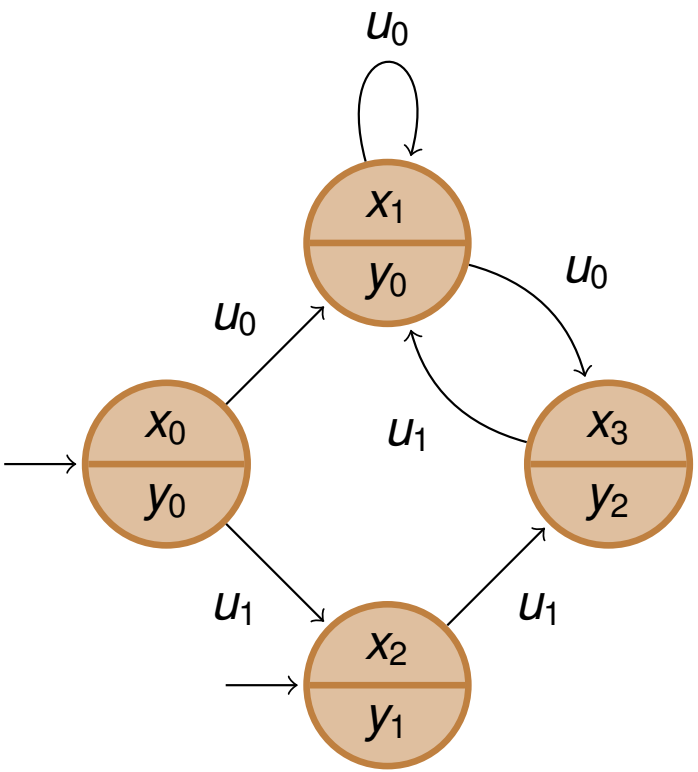
CyPhyLab UCLA



$$X = \{x_0, x_1, x_2, x_3\}$$



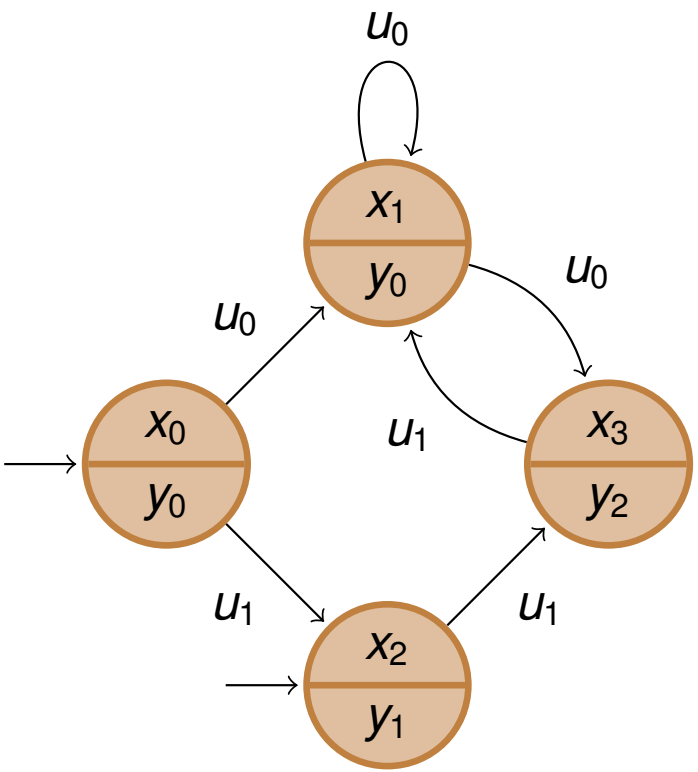
$$X = \{x_0, x_1, x_2, x_3\}$$
$$X_0 = \{x_0, x_1\}$$



$$
\begin{aligned}
X &= \{x_0, x_1, x_2, x_3\} \\
X_0 &= \{x_0, x_1\} \\
U &= \{u_0, u_1\}
\end{aligned}
$$

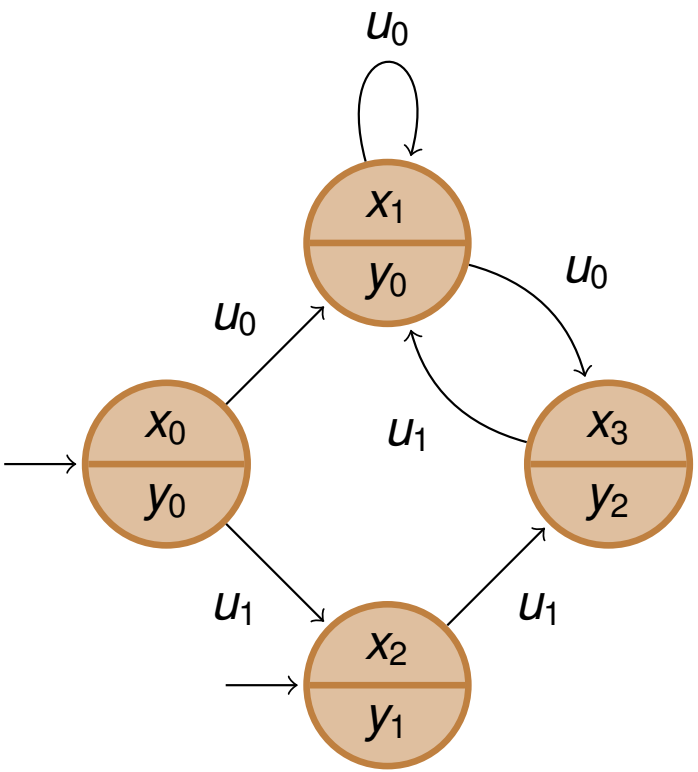
Systems

Definitions



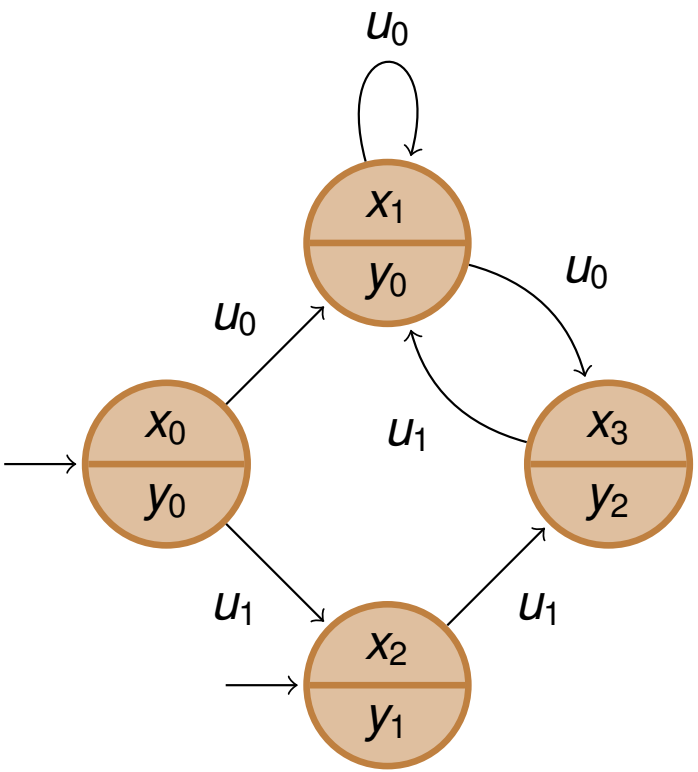
$$
\begin{aligned}
X &= \{x_0, x_1, x_2, x_3\} \\
X_0 &= \{x_0, x_1\} \\
U &= \{u_0, u_1\} \\
\longrightarrow &= \{(x_0, u_0, x_1), (x_0, u_1, x_2), (x_1, u_0, x_1), \\
&\quad (x_1, u_0, x_3), (x_2, u_1, x_3), (x_3, u_1, x_1)\}
\end{aligned}
$$

CyPhyLab UCLA



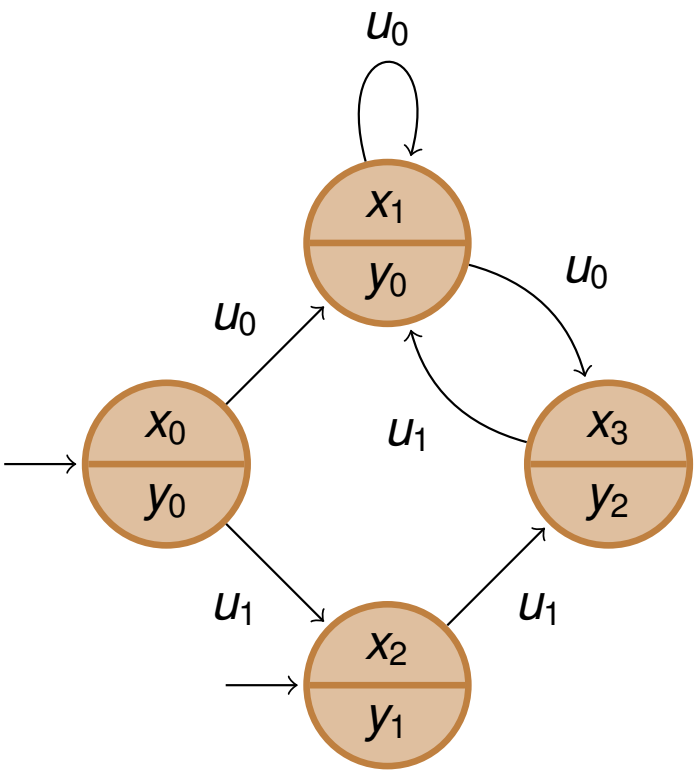
$$
\begin{aligned}
X &= \{x_0, x_1, x_2, x_3\} \\
X_0 &= \{x_0, x_1\} \\
U &= \{u_0, u_1\} \\
\longrightarrow &= \{(x_0, u_0, x_1), (x_0, u_1, x_2), (x_1, u_0, x_1), \\
&\quad (x_1, u_0, x_3), (x_2, u_1, x_3), (x_3, u_1, x_1)\} \\
Y &= \{y_0, y_1, y_2\}
\end{aligned}
$$

CyPhyLab UCLA

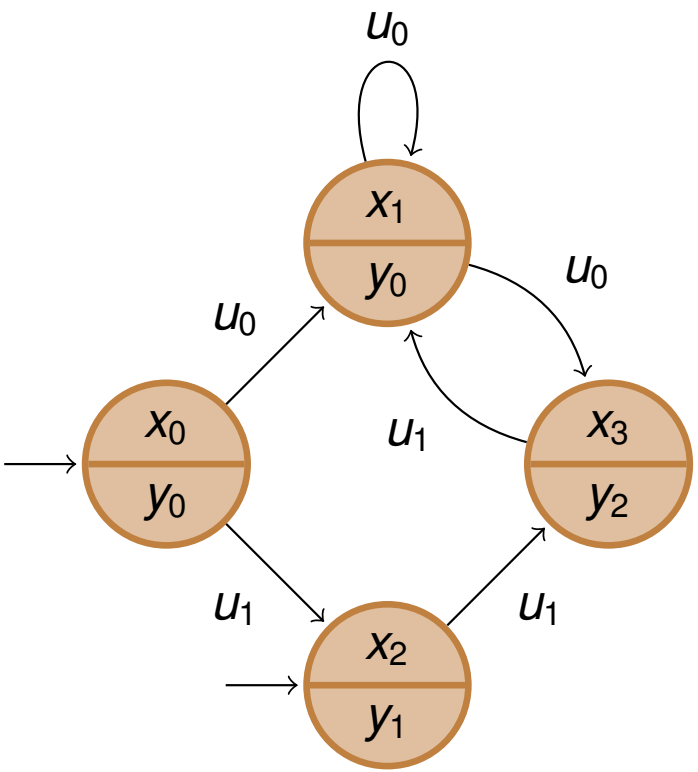


$$
\begin{aligned}
X &= \{x_0, x_1, x_2, x_3\} \\
X_0 &= \{x_0, x_1\} \\
U &= \{u_0, u_1\} \\
\longrightarrow &= \{(x_0, u_0, x_1), (x_0, u_1, x_2), (x_1, u_0, x_1), \\
&\quad (x_1, u_0, x_3), (x_2, u_1, x_3), (x_3, u_1, x_1)\} \\
Y &= \{y_0, y_1, y_2\} \\
H(x_0) &= y_0, \quad H(x_1) = y_0, \quad H(x_2) = y_1 \\
H(x_3) &= y_2.
\end{aligned}
$$

CyPhyLab UCLA

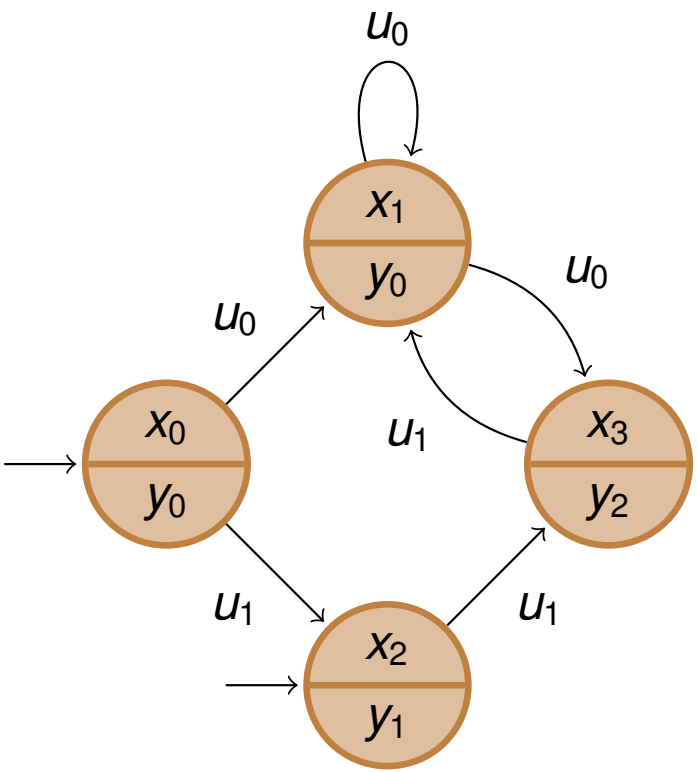


Consider the transition $x_0 \xrightarrow{u_0} x_1$.



Consider the transition $x_0 \xrightarrow{u_0} x_1$.

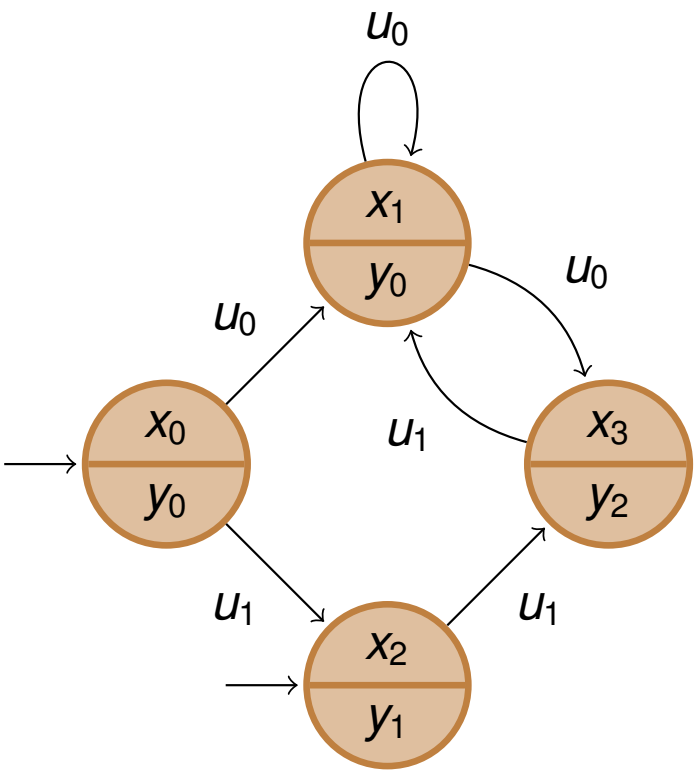
State x_1 is called a u_0-*successor*, or simply *successor*, of state x_0. Similarly, x_0 is called a u_0-*predecessor*, or *predecessor*, of state x_1.



Consider the transition $x_0 \xrightarrow{u_0} x_1$.

State x_1 is called a u_0-*successor*, or simply *successor*, of state x_0. Similarly, x_0 is called a u_0-*predecessor*, or *predecessor*, of state x_1.

We denote the set of u-successors of a state x by $\mathrm{Post}_u(x)$.

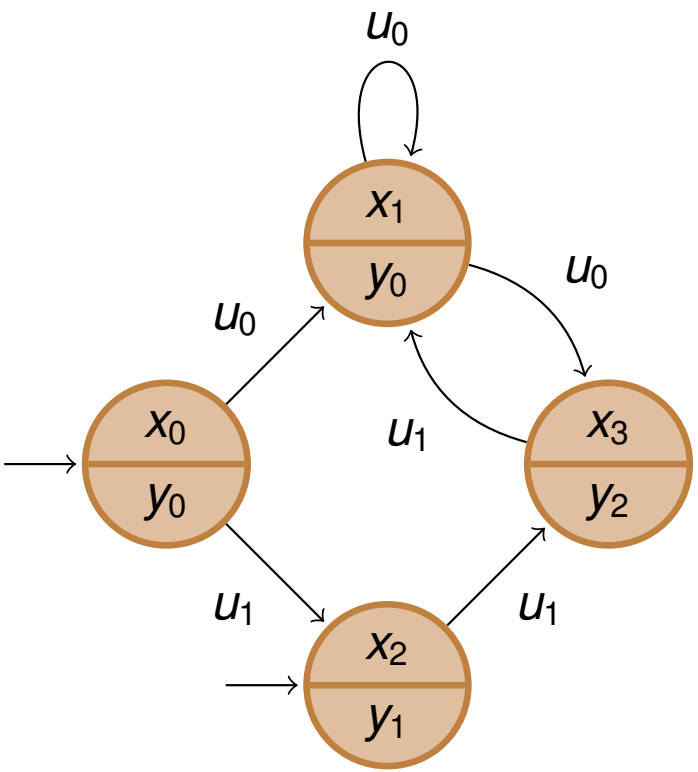


Consider the transition $x_0 \xrightarrow{u_0} x_1$.

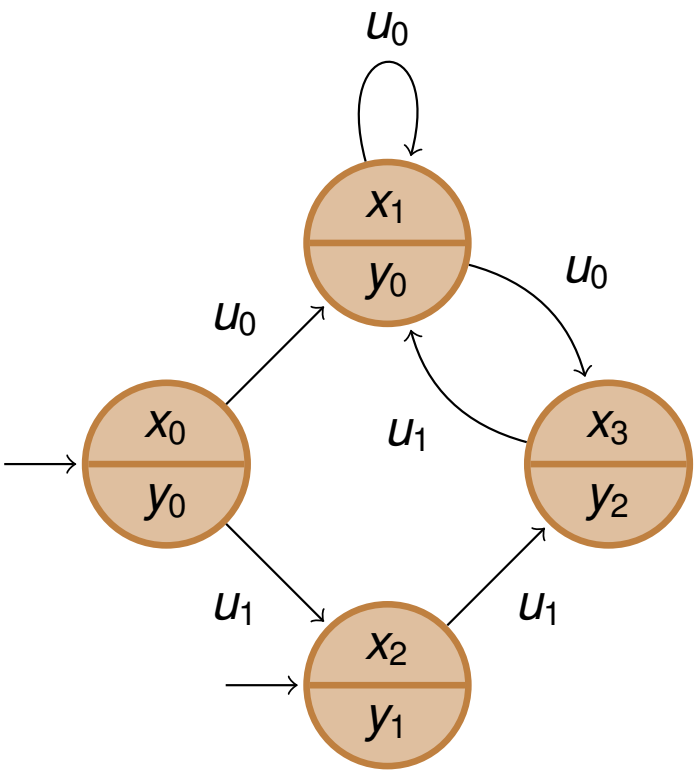
State x_1 is called a u_0-*successor*, or simply *successor*, of state x_0. Similarly, x_0 is called a u_0-*predecessor*, or *predecessor*, of state x_1.

We denote the set of u-successors of a state x by $\mathrm{Post}_u(x)$.

Since $\mathrm{Post}_u(x)$ may be empty, we denote by $U(x)$ the set of inputs $u \in U$ for which $\mathrm{Post}_u(x)$ is nonempty.



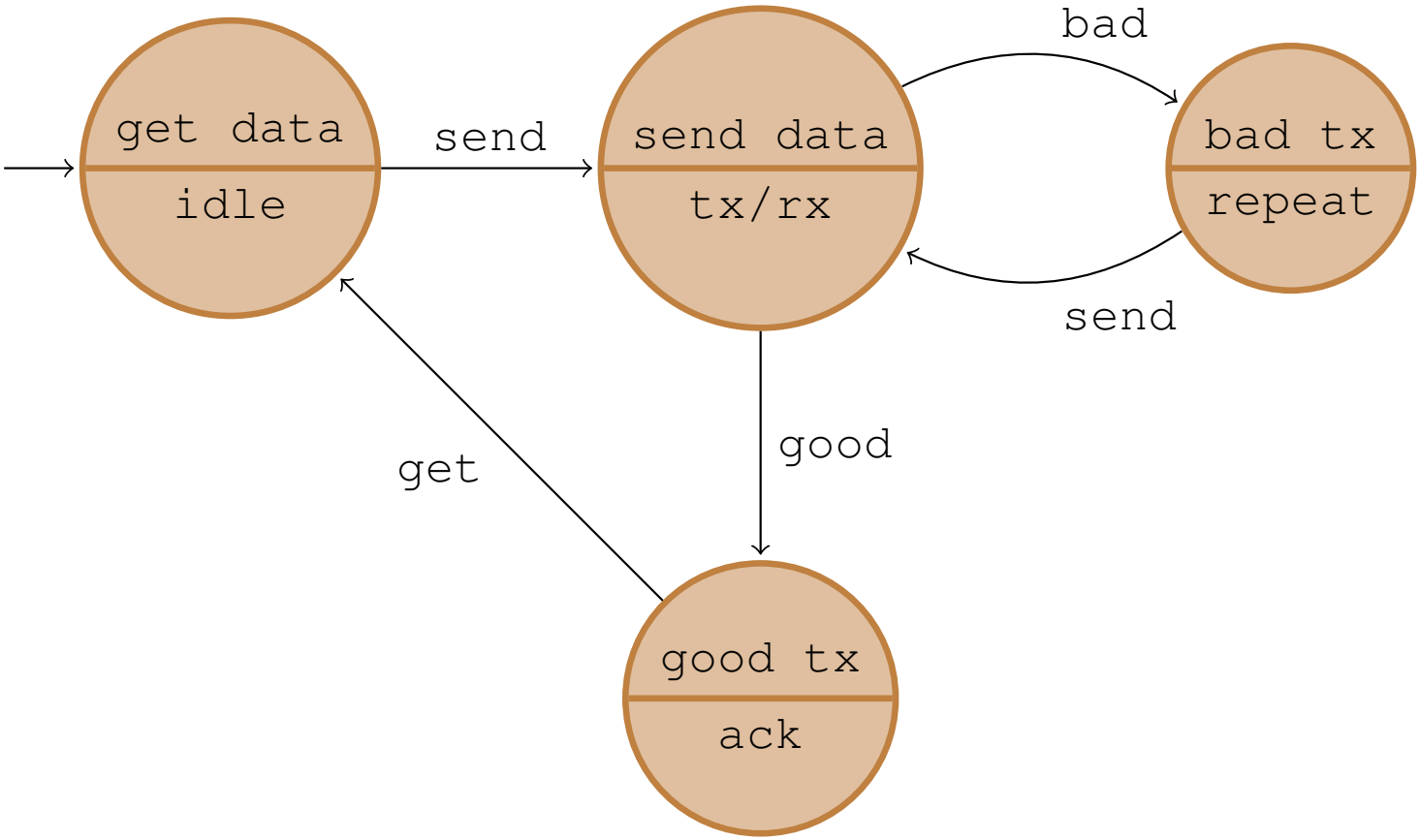
This system is nondeterministic since there are two distinct u_0-successors of x_1, i.e. $\mathrm{Post}_{u_0}(x_1) = \{x_1, x_3\}$.



This system is nondeterministic since there are two distinct u_0-successors of x_1, i.e. $\mathrm{Post}_{u_0}(x_1) = \{x_1, x_3\}$.

This system is non-blocking since every state has at least one successor, i.e. $U(x) \neq \varnothing$ for every $x \in X$.

A simple communication protocol (transmission side).



The computation of averages is a problem that occurs frequently in applications.

Suppose that we want to compute the average of a stream of numbers but we do not know a priori the length of the stream.

The computation of averages is a problem that occurs frequently in applications.

Suppose that we want to compute the average of a stream of numbers but we do not know a priori the length of the stream.

$$
\begin{aligned}
&\texttt{x} := 0\\
&\texttt{n} := 0\\
&\texttt{While}(\textbf{true})\\
&\quad \{\\
&\quad \texttt{y} := \texttt{read}(\texttt{input})\\
&\quad \texttt{x} := \texttt{x}\frac{\texttt{n}}{\texttt{n}+1} + \texttt{y}\frac{1}{\texttt{n}+1}\\
&\quad \texttt{n} := \texttt{n}+1\\
&\quad \}
\end{aligned}
$$

The variable *y* contains the latest received number and the variable *x* contains the average of the numbers that have been received so far.

Assume now that we are interested in knowing if x is smaller, equal, or greater than 1 when y is restricted to assume values in the set $\{1, 2\}$.

$$\texttt{x} := 0$$

$$\texttt{n} := 0$$

$$\texttt{While}(\textbf{true})$$

$$\{$$

$$\texttt{y} := \texttt{read}(\texttt{input})$$

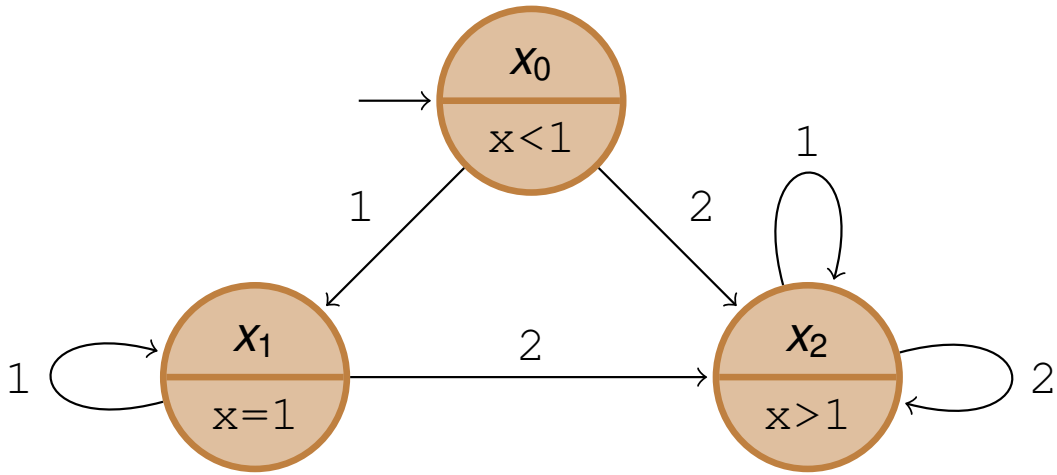
$$\texttt{x} := \texttt{x}\frac{\texttt{n}}{\texttt{n}+1} + \texttt{y}\frac{1}{\texttt{n}+1}$$

$$\texttt{n} := \texttt{n}+1$$

$$\}$$

Assume now that we are interested in knowing if x is smaller, equal, or greater than 1 when y is restricted to assume values in the set $\{1, 2\}$.

```
x := 0
n := 0
While(true)
    {
    y := read(input)
    x := x n/(n+1) + y 1/(n+1)
    n := n + 1
    }
```



Dynamical systems evolving in continuous-time can also be modeled as systems. Consider Euler's equations for the rotational dynamics of a rigid body around its center of mass:

$$\frac{d}{dt}\xi_1 = \frac{I_2 - I_3}{I_1}\xi_2\xi_3 \quad (1)$$

$$\frac{d}{dt}\xi_2 = \frac{I_3 - I_1}{I_2}\xi_3\xi_1 \quad (2)$$

$$\frac{d}{dt}\xi_3 = \frac{I_1 - I_2}{I_3}\xi_1\xi_2 \quad (3)$$

where $\xi = (\xi_1, \xi_2, \xi_3)$ is the body angular velocity and $I_1, I_2, I_3 \in \mathbb{R}$ are the principal moments of inertia.

The rigid body can be described by the system $S = (X, X_0, U, \longrightarrow, Y, H)$ consisting of:

- $X = \mathbb{R}^3$;
- $X_0 = X$;
- $U = \mathbb{R}_0^+$;
- $x \xrightarrow{\tau} x'$ if there exists a solution ξ to equations (1) through (3) satisfying $\xi(0) = x$ and $\xi(\tau) = x'$;
- $Y = X$;
- $H = 1_X$.

The rigid body can be described by the system $S = (X, X_0, U, \longrightarrow, Y, H)$ consisting of:

- $X = \mathbb{R}^3$;
- $X_0 = X$;
- $U = \mathbb{R}_0^+$;
- $x \xrightarrow{\tau} x'$ if there exists a solution ξ to equations (1) through (3) satisfying $\xi(0) = x$ and $\xi(\tau) = x'$;
- $Y = X$;
- $H = 1_X$.

Several other classes of systems can be conveniently described by the proposed notion of system. Examples include timed-automata, switched systems, hybrid systems, and many others.

Given a system $S = (X, X_0, U, \longrightarrow, Y, H)$ and a state $x \in X$, a *finite internal behavior* generated from x is a finite sequence of transitions:

$$x_0 \xrightarrow{u_0} x_1 \xrightarrow{u_1} x_2 \xrightarrow{u_2} \ldots \xrightarrow{u_{n-2}} x_{n-1} \xrightarrow{u_{n-1}} x_n$$

such that $x_0 = x$ and $x_i \xrightarrow{u_i} x_{i+1}$ for all $0 \leq i < n$.

An internal behavior generated from x is *initialized* if $x \in X_0$.

Given a system $S = (X, X_0, U, \longrightarrow, Y, H)$ and a state $x \in X$, a *finite internal behavior* generated from x is a finite sequence of transitions:

$$x_0 \xrightarrow{u_0} x_1 \xrightarrow{u_1} x_2 \xrightarrow{u_2} \dots \xrightarrow{u_{n-2}} x_{n-1} \xrightarrow{u_{n-1}} x_n$$

such that $x_0 = x$ and $x_i \xrightarrow{u_i} x_{i+1}$ for all $0 \leq i < n$.

An internal behavior generated from x is *initialized* if $x \in X_0$.

In some cases, a finite internal behavior can be extended to an infinite internal behavior. An *infinite internal behavior* generated from x is an infinite sequence:

$$x_0 \xrightarrow{u_0} x_1 \xrightarrow{u_1} x_2 \xrightarrow{u_2} x_3 \xrightarrow{u_3} \dots$$

that satisfies $x_0 = x$ and $x_i \xrightarrow{u_i} x_{i+1}$ for all $i \in \mathbb{N}_0$. An infinite internal behavior generated from x is called *initialized* if $x \in X_0$.

In nonblocking systems, every finite internal behavior can be extended to an infinite internal behavior.

Through the output map, every internal behavior:

$$x_0 \xrightarrow{u_0} x_1 \xrightarrow{u_1} x_2 \xrightarrow{u_2} \cdots \xrightarrow{u_{n-2}} x_{n-1} \xrightarrow{u_{n-1}} x_n$$

defines an external behavior:

$$y_0 \longrightarrow y_1 \longrightarrow y_2 \longrightarrow \cdots \longrightarrow y_{n-1} \longrightarrow y_n$$

with $H(x_i) = y_i \in Y$ for all $0 \leq i \leq n$.

Through the output map, every internal behavior:

$$x_0 \xrightarrow{u_0} x_1 \xrightarrow{u_1} x_2 \xrightarrow{u_2} \dots \xrightarrow{u_{n-2}} x_{n-1} \xrightarrow{u_{n-1}} x_n$$

defines an external behavior:

$$y_0 \longrightarrow y_1 \longrightarrow y_2 \longrightarrow \dots \longrightarrow y_{n-1} \longrightarrow y_n$$

with $H(x_i) = y_i \in Y$ for all $0 \leq i \leq n$.

The set of external behaviors that are defined by internal behaviors generated from state x is denoted by $\mathcal{B}_x(S)$ and is called the *external behavior* from state x.

Definition (Finite External Behavior)

The *finite external behavior* generated by a system S, denoted by $\mathcal{B}(S)$, is defined by:

$$\mathcal{B}(S) = \bigcup_{x \in X_0} \mathcal{B}_x(S).$$

An infinite internal behavior from x:

$$x_0 \xrightarrow{u_0} x_1 \xrightarrow{u_1} x_2 \xrightarrow{u_2} x_3 \xrightarrow{u_3} \dots$$

defines an infinite external behavior:

$$y_0 \longrightarrow y_1 \longrightarrow y_2 \longrightarrow y_3 \longrightarrow \dots$$

corresponding to the infinite sequence of outputs with $H(x_i) = y_i$ for all $i \in \mathbb{N}_0$.

An infinite internal behavior from x:

$$x_0 \xrightarrow{u_0} x_1 \xrightarrow{u_1} x_2 \xrightarrow{u_2} x_3 \xrightarrow{u_3} \dots$$

defines an infinite external behavior:

$$y_0 \longrightarrow y_1 \longrightarrow y_2 \longrightarrow y_3 \longrightarrow \dots$$

corresponding to the infinite sequence of outputs with $H(x_i) = y_i$ for all $i \in \mathbb{N}_0$.

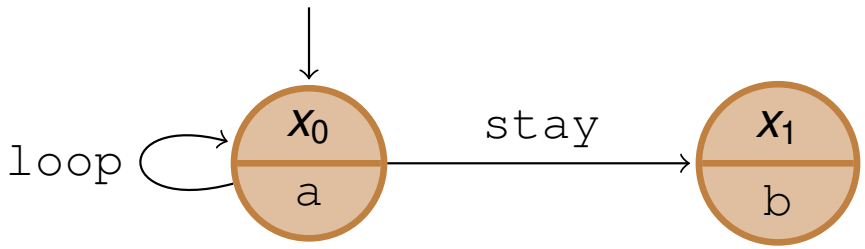
The set of all infinite external behaviors that are generated from x is denoted by $\mathcal{B}^\omega_x(S)$ and called the *infinite external behavior* from state x.

Definition (Infinite External Behavior)

The *infinite external behavior* generated by a system S, denoted by $\mathcal{B}^\omega(S)$, is defined by:

$$\mathcal{B}^\omega(S) = \bigcup_{x \in X_0} \mathcal{B}^\omega_x(S).$$

If a system S is non-blocking, then $\mathcal{B}^\omega(S)$ is nonempty. However, $\mathcal{B}^\omega(S)$ may be nonempty even if S is a blocking system.



The infinite external behavior $\texttt{aaaaa}\ldots$ belongs to $\mathcal{B}^\omega(S)$ although S is blocking since the state x_1 has no successors.

Discrete Synthesis

Safety games

The first class of synthesis problems that we consider are safety games.

We are given a system S and a set of safe outputs $W \subseteq Y$. The objective is to synthesize a controller C such that S_C is nonblocking and $\mathcal{B}^\omega(S_C) \subseteq W^\omega$.

Discrete Synthesis

Safety games

The first class of synthesis problems that we consider are safety games.

We are given a system S and a set of safe outputs $W \subseteq Y$. The objective is to synthesize a controller C such that S_C is nonblocking and $\mathcal{B}^\omega(S_C) \subseteq W^\omega$.

Several concrete requirements can be formulated as safety specifications: avoid buffer overflows, avoid collision with obstacles, avoid leaving operational regions, etc.

CyPhyLab UCLA

Discrete Synthesis

Safety games

The first class of synthesis problems that we consider are safety games.

We are given a system S and a set of safe outputs $W \subseteq Y$. The objective is to synthesize a controller C such that S_C is nonblocking and $\mathcal{B}^\omega(S_C) \subseteq W^\omega$.

Several concrete requirements can be formulated as safety specifications: avoid buffer overflows, avoid collision with obstacles, avoid leaving operational regions, etc.

How do we define a controller C? How does C control a system S?

Discrete Synthesis

Safety games

The first class of synthesis problems that we consider are safety games.

We are given a system S and a set of safe outputs $W \subseteq Y$. The objective is to synthesize a controller C such that S_C is nonblocking and $\mathcal{B}^\omega(S_C) \subseteq W^\omega$.

Several concrete requirements can be formulated as safety specifications: avoid buffer overflows, avoid collision with obstacles, avoid leaving operational regions, etc.

How do we define a controller C? How does C control a system S?

Definition

A controller C for a system $S = (X, X_0, U, \longrightarrow, Y, H)$ is a map $C : X \to 2^U$. The result of controlling system S with controller C is described by the system $S_C = (X, X_{0C}, U, \xrightarrow[C]{}, Y, H)$ with transition relation $\xrightarrow[C]{}$ defined by:

$$x \xrightarrow[C]{u} x' \quad \text{if} \quad x \xrightarrow{u} x' \wedge u \in C(x).$$

CyPhyLab UCLA

Discrete Synthesis

Safety games

The first class of synthesis problems that we consider are safety games.

We are given a system S and a set of safe outputs $W \subseteq Y$. The objective is to synthesize a controller C such that S_C is nonblocking and $\mathcal{B}^\omega(S_C) \subseteq W^\omega$.

Several concrete requirements can be formulated as safety specifications: avoid buffer overflows, avoid collision with obstacles, avoid leaving operational regions, etc.

How do we define a controller C? How does C control a system S?

Definition

A controller C for a system $S = (X, X_0, U, \longrightarrow, Y, H)$ is a map $C : X \to 2^U$. The result of controlling system S with controller C is described by the system $S_C = (X, X_{0C}, U, \xrightarrow[C]{}, Y, H)$ with transition relation $\xrightarrow[C]{}$ defined by:

$$x \xrightarrow[C]{u} x' \quad \text{if} \quad x \xrightarrow{u} x' \wedge u \in C(x).$$

In other words, we restrict $U(x)$ to $U(x) \cap C(x)$.

CyPhyLab UCLA

Definition (Safety game)

Let S be a system with $Y = X$ and $H = 1_X$, and let $W \subseteq X$ be a set of safe states. The *safety game* for system S and specification set W asks for the existence of a controller C such that:

1. S_C is nonblocking;
2. $\varnothing \neq \mathcal{B}^\omega(S_C) \subseteq W^\omega$.

A safety game is said to be solvable when S_C exists.

The requirement $Y = X$ and $H = 1_X$ is made without loss of generality since the general case where $Y \neq X$ can be reduced to this one.

Safety games can be solved by constructing a suitable operator:

$$F_W : 2^X \to 2^X$$

for any specification set $W \subseteq X$:

$$F_W(Z) = \{x \in Z \mid x \in W \text{ and } \exists u \in U(x) \quad \varnothing \neq \text{Post}_u(x) \subseteq Z\}.$$

Safety games can be solved by constructing a suitable operator:

$$F_W : 2^X \to 2^X$$

for any specification set $W \subseteq X$:

$$F_W(Z) = \{x \in Z \mid x \in W \text{ and } \exists u \in U(x) \quad \varnothing \neq \text{Post}_u(x) \subseteq Z\}.$$

A fixed-point for F_W is a set $Z \subseteq X$ satisfying $F_W(Z) = Z$. Such fixed-point defines a set Z of states from which it is possible to control system S so as to remain in W.

Proposition

Let S be a system with $Y = X$ and $H = 1_X$, and let $W \subseteq X$ be a set of safe states. The safety game for system S and specification set W is solvable iff the maximal fixed-point Z of F_W satisfies $X_0 \subseteq Z$.

How do we construct the controller from a fixed-point Z of F_W satisfying $X_0 \subseteq Z$?

How do we construct the controller from a fixed-point Z of F_W satisfying $X_0 \subseteq Z$?

We can take:

$$C(x) = \{u \in U(x) \mid \mathrm{Post}_u(x) \subseteq Z\}. \tag{4}$$

How do we construct the controller from a fixed-point Z of F_W satisfying $X_0 \subseteq Z$?

We can take:

$$C(x) = \{u \in U(x) \mid \mathrm{Post}_u(x) \subseteq Z\}. \tag{4}$$

Will this controller work?

How do we construct the controller from a fixed-point Z of F_W satisfying $X_0 \subseteq Z$?

We can take:

$$C(x) = \{u \in U(x) \mid \text{Post}_u(x) \subseteq Z\}. \tag{4}$$

Will this controller work?

Since $X_0 \subseteq Z$ we can start at a state in X_0. Then, by using any input given by C we move to a new state still in Z. Hence, we can stay in $Z \subseteq W$ forever.

How do we construct the controller from a fixed-point Z of F_W satisfying $X_0 \subseteq Z$?

We can take:

$$C(x) = \{u \in U(x) \mid \text{Post}_u(x) \subseteq Z\}. \tag{4}$$

Will this controller work?

Since $X_0 \subseteq Z$ we can start at a state in X_0. Then, by using any input given by C we move to a new state still in Z. Hence, we can stay in $Z \subseteq W$ forever.

If we have the possibility to set the initial state, then we can relax $X_0 \subseteq Z$ to $Z \cap X_0 \neq \varnothing$ so that there exists at least one initial state belonging to Z.

A complete characterization of the solutions to safety games can now be obtained by noting that the maximal fixed-point of F_W can be obtained by iteration.

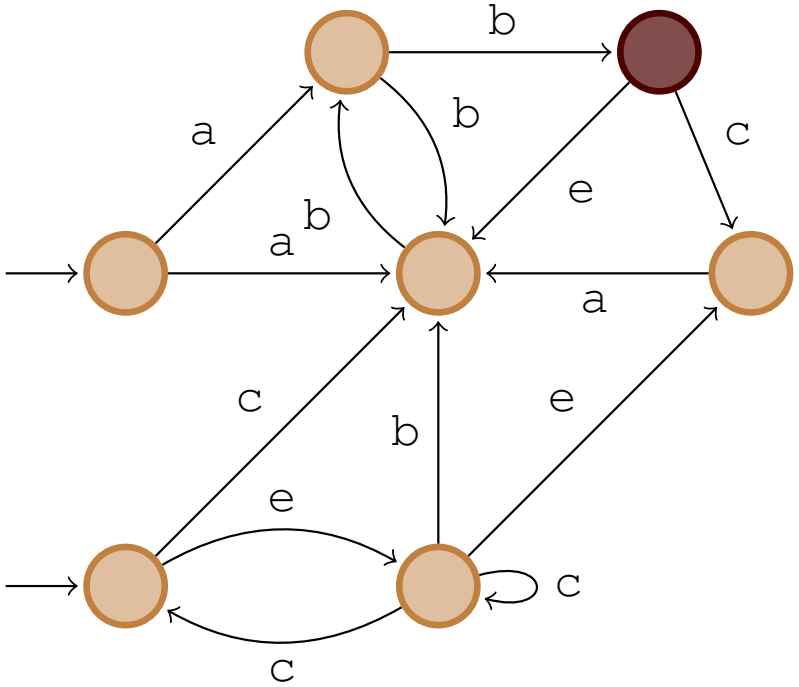
Theorem

Let S be a system with $Y = X$ and $H = 1_X$, and let $W \subseteq X$ be a set of safe states. The safety game for system S and specification set W is solvable iff the maximal fixed-point Z of the operator F_W satisfies $X_0 \subseteq Z$. Moreover, Z can be obtained as:

$$Z = \lim_{i \to \infty} F_W^i(X).$$

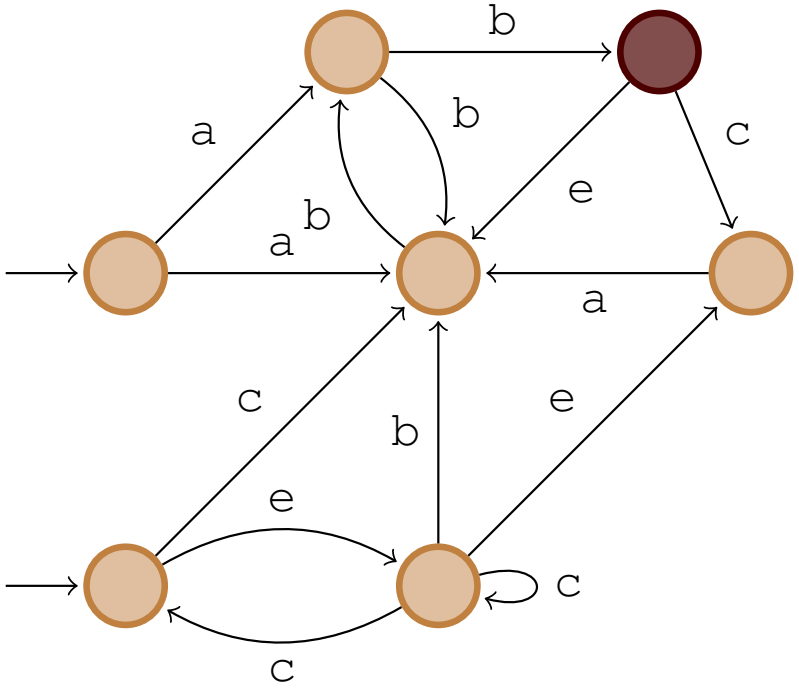
When $X_0 \subseteq Z$, a solution to the safety game is given by the controller (4).

Example: synthesize a controller forcing the closed-loop system not to enter the darker state.



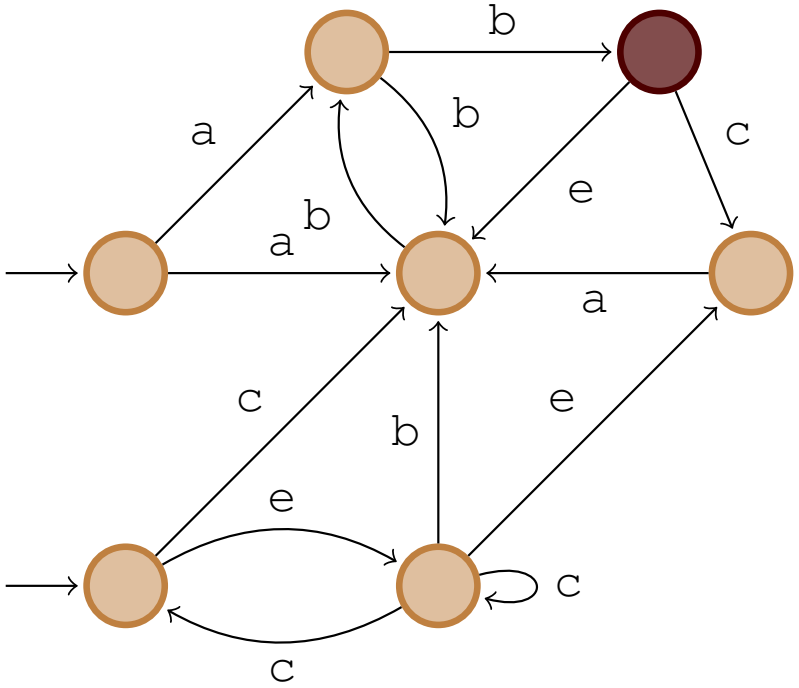
Example: synthesize a controller forcing the closed-loop system not to enter the darker state.

$$F_W(Z) = \{x \in Z \mid x \in W \text{ and } \exists u \in U(x) \quad \varnothing \neq \text{Post}_u(x) \subseteq Z\}$$



Example: synthesize a controller forcing the closed-loop system not to enter the darker state.

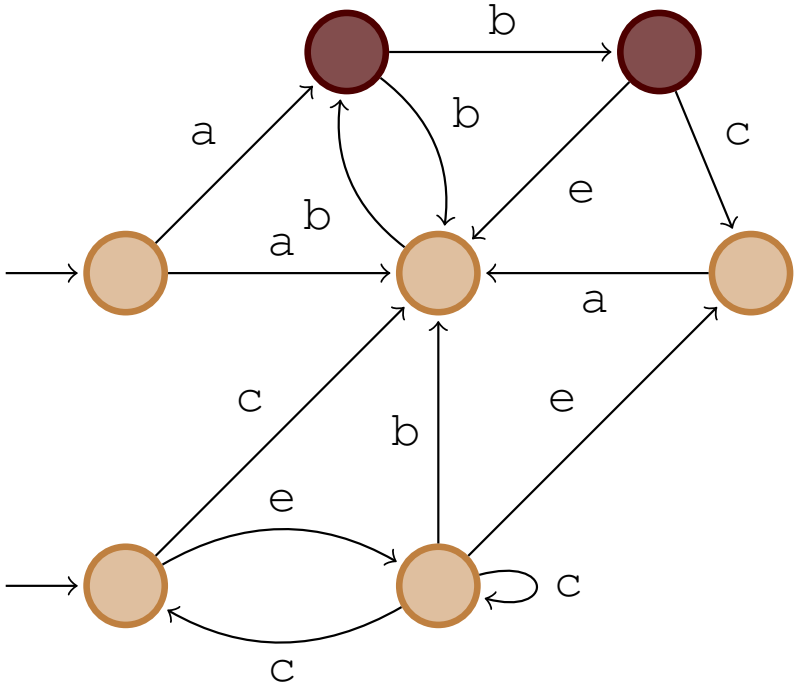
$$F_W(Z) = \{x \in Z \mid x \in W \text{ and } \exists u \in U(x) \quad \varnothing \neq \text{Post}_u(x) \subseteq Z\}$$



$F_W^1(X)$

Example: synthesize a controller forcing the closed-loop system not to enter the darker state.

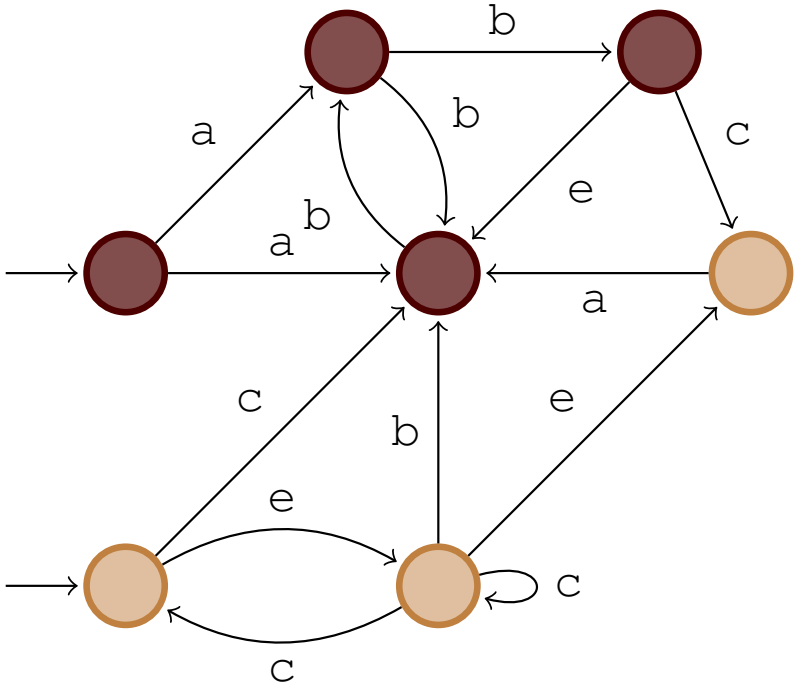
$$F_W(Z) = \{x \in Z \mid x \in W \text{ and } \exists u \in U(x) \quad \varnothing \neq \text{Post}_u(x) \subseteq Z\}$$



$F_W^2(X)$

Example: synthesize a controller forcing the closed-loop system not to enter the darker state.

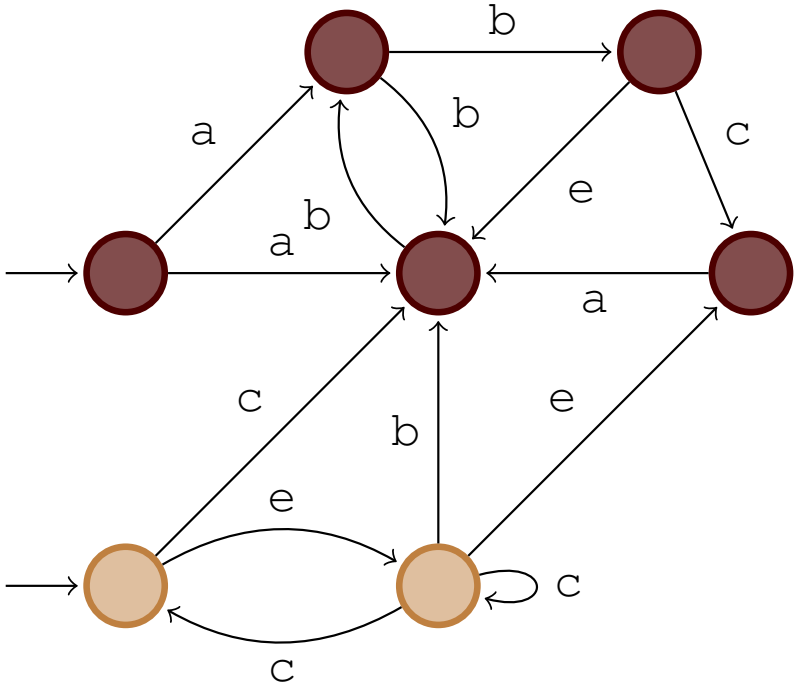
$$F_W(Z) = \{x \in Z \mid x \in W \text{ and } \exists u \in U(x) \quad \varnothing \neq \text{Post}_u(x) \subseteq Z\}$$



$F_W^3(X)$

Example: synthesize a controller forcing the closed-loop system not to enter the darker state.

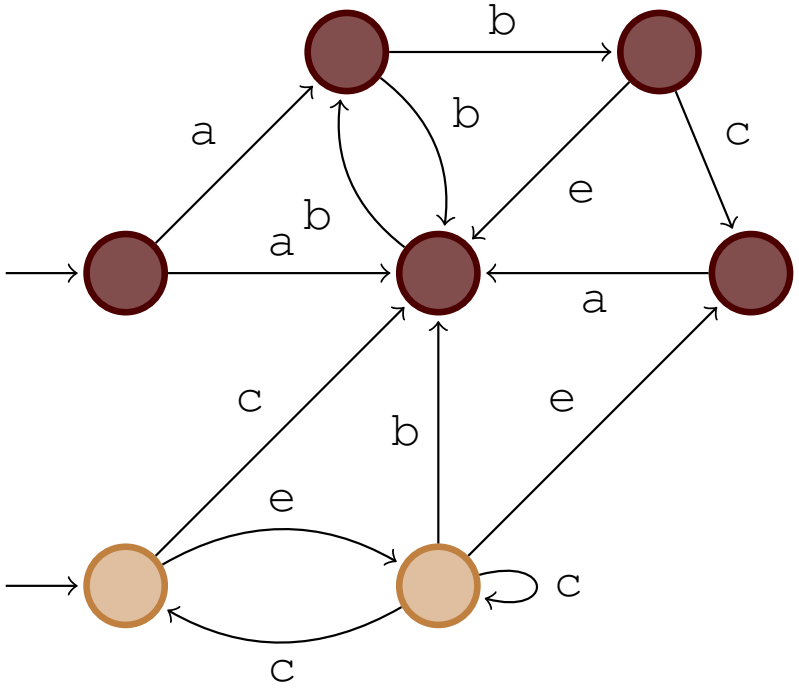
$$F_W(Z) = \{x \in Z \mid x \in W \text{ and } \exists u \in U(x) \quad \varnothing \neq \text{Post}_u(x) \subseteq Z\}$$



$F_W^4(X)$

Example: synthesize a controller forcing the closed-loop system not to enter the darker state.

$$F_W(Z) = \{x \in Z \mid x \in W \text{ and } \exists u \in U(x) \quad \varnothing \neq \text{Post}_u(x) \subseteq Z\}$$



$$F_W^5(X)$$

If we use the maximal fixed point of F_W then we have the following maximality property for the corresponding controller:

Proposition

Let S be a system with $Y = X$ and $H = 1_X$, and let $W \subseteq X$ be a set of safe states. For any controller D solving the safety game for system S and specification set W and for any $x \in Z$ we have:

$$D(x) \subseteq C(x)$$

where C is the controller (4) defined by the maximal fixed-point Z of F_W.

If we use the maximal fixed point of F_W then we have the following maximality property for the corresponding controller:

Proposition

Let S be a system with $Y = X$ and $H = 1_X$, and let $W \subseteq X$ be a set of safe states. For any controller D solving the safety game for system S and specification set W and for any $x \in Z$ we have:

$$D(x) \subseteq C(x)$$

where C is the controller (4) defined by the maximal fixed-point Z of F_W.

What do we do when $H \neq 1_X$?

If we use the maximal fixed point of F_W then we have the following maximality property for the corresponding controller:

Proposition

Let S be a system with $Y = X$ and $H = 1_X$, and let $W \subseteq X$ be a set of safe states. For any controller D solving the safety game for system S and specification set W and for any $x \in Z$ we have:

$$D(x) \subseteq C(x)$$

where C is the controller (4) defined by the maximal fixed-point Z of F_W.

What do we do when $H \neq 1_X$?

It suffices to consider a new safe set $W' \subseteq X$ defined by $W' = H^{-1}(W)$ and to synthesize a controller for this new set of safe states.

The second class of control problems that we consider are reachability games. We are given a system S and a set of outputs $W \subseteq Y$ to be reached.

The second class of control problems that we consider are reachability games. We are given a system S and a set of outputs $W \subseteq Y$ to be reached.

Definition (Reachability game)

Let S be a system satisfying $Y = X$ and $H = 1_X$, and let $W \subseteq X$ be a set of states. The *reachability game* for system S and specification set W asks for the existence of a controller C such that for every maximal behavior $y \in \mathcal{B}(S_C) \cup \mathcal{B}^\omega(S_C)$ there exists $k \in \mathbb{N}_0$ such that $y(k) = y_k \in W$. A reachability game is said to be solvable when S_C exists.

No longer asking for the controlled system S_C system to be nonblocking.

The second class of control problems that we consider are reachability games. We are given a system S and a set of outputs $W \subseteq Y$ to be reached.

Definition (Reachability game)

Let S be a system satisfying $Y = X$ and $H = 1_X$, and let $W \subseteq X$ be a set of states. The *reachability game* for system S and specification set W asks for the existence of a controller C such that for every maximal behavior $y \in \mathcal{B}(S_C) \cup \mathcal{B}^\omega(S_C)$ there exists $k \in \mathbb{N}_0$ such that $y(k) = y_k \in W$. A reachability game is said to be solvable when S_C exists.

No longer asking for the controlled system S_C system to be nonblocking.

As in the case of safety games, it suffices to consider the case where $Y = X$ and $H = 1_X$.

The solution of reachability games is also based on a suitable operator. For any $W \subseteq X$ we can define the operator:

$$G_W : 2^X \to 2^X$$

by:

$$G_W(Z) \quad = \quad \{x \in X \mid x \in W \text{ or } \exists u \in U(x) \quad \varnothing \neq \text{Post}_u(x) \subseteq Z\}.$$

The solution of reachability games is also based on a suitable operator. For any $W \subseteq X$ we can define the operator:

$$G_W : 2^X \to 2^X$$

by:

$$G_W(Z) \quad = \quad \{x \in X \mid x \in W \text{ or } \exists u \in U(x) \quad \varnothing \neq \text{Post}_u(x) \subseteq Z\}.$$

Theorem

Let S be a system with $Y = X$ and $H = 1_X$, and let $W \subseteq X$ be a set of states. The reachability game for S and specification set W is solvable iff the minimal fixed-point Z of the operator G_W satisfies $X_0 \subseteq Z$. Moreover, Z can be obtained as:

$$Z = \lim_{i \to \infty} G_W^i(\varnothing).$$

The solution of reachability games is also based on a suitable operator. For any $W \subseteq X$ we can define the operator:

$$G_W : 2^X \to 2^X$$

by:

$$G_W(Z) \quad = \quad \{x \in X \mid x \in W \text{ or } \exists u \in U(x) \quad \varnothing \neq \text{Post}_u(x) \subseteq Z\}.$$

Theorem

Let S be a system with $Y = X$ and $H = 1_X$, and let $W \subseteq X$ be a set of states. The reachability game for S and specification set W is solvable iff the minimal fixed-point Z of the operator G_W satisfies $X_0 \subseteq Z$. Moreover, Z can be obtained as:

$$Z = \lim_{i \to \infty} G_W^i(\varnothing).$$

However, there is no maximal controller.

Several different controllers solving the reachability game can be constructed from a fixed-point Z of G_W for which $X_0 \subseteq Z$.

Several different controllers solving the reachability game can be constructed from a fixed-point Z of G_W for which $X_0 \subseteq Z$. Here is one such possibility:

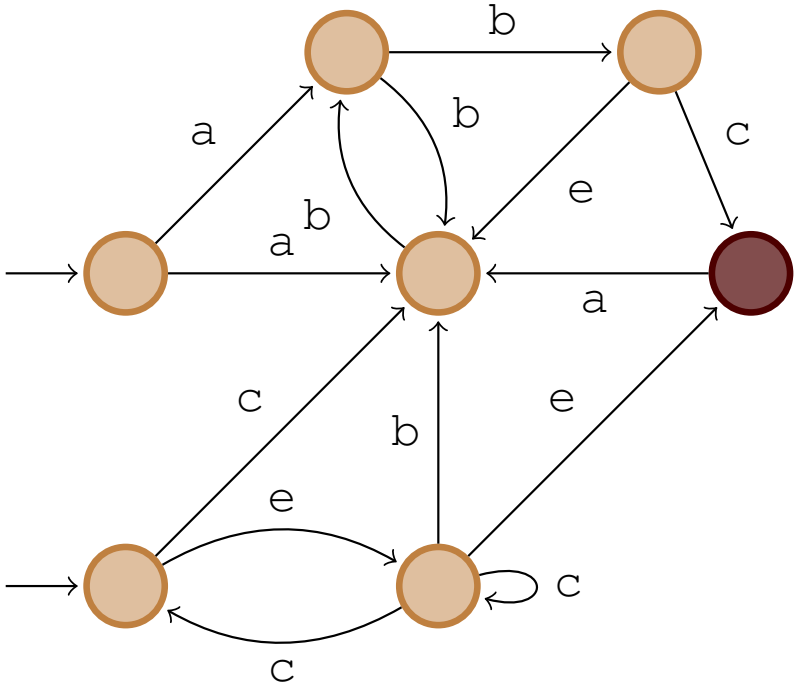
$$C(x) = \{u \in U(x) \mid \exists k \in \mathbb{N} \text{ such that } x \notin G_W^k(\varnothing) \text{ and } \varnothing \neq \mathrm{Post}_u(x) \subseteq G_W^k(\varnothing)\} \quad (5)$$

Several different controllers solving the reachability game can be constructed from a fixed-point Z of G_W for which $X_0 \subseteq Z$. Here is one such possibility:

$$C(x) = \{u \in U(x) \mid \exists k \in \mathbb{N} \text{ such that } x \notin G_W^k(\varnothing) \text{ and } \varnothing \neq \mathrm{Post}_u(x) \subseteq G_W^k(\varnothing)\} \quad (5)$$

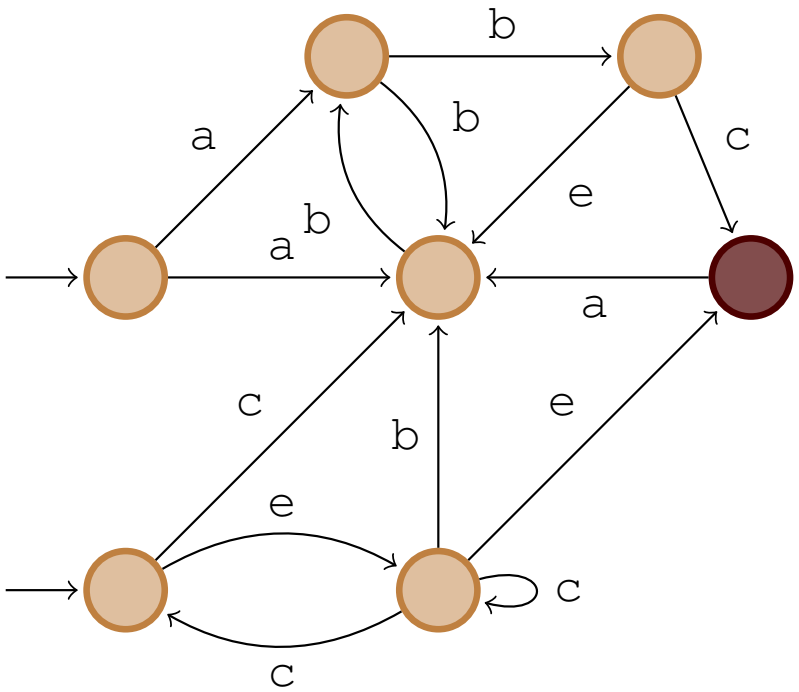
This consists of the minimum-time solution to the reachability problem. The input u belongs to $C(x)$ if there exists a k such that W can be reached from x in k steps but not in $k-1$ steps.

Consider the following finite-state system and assume that W consists of the single darker state.



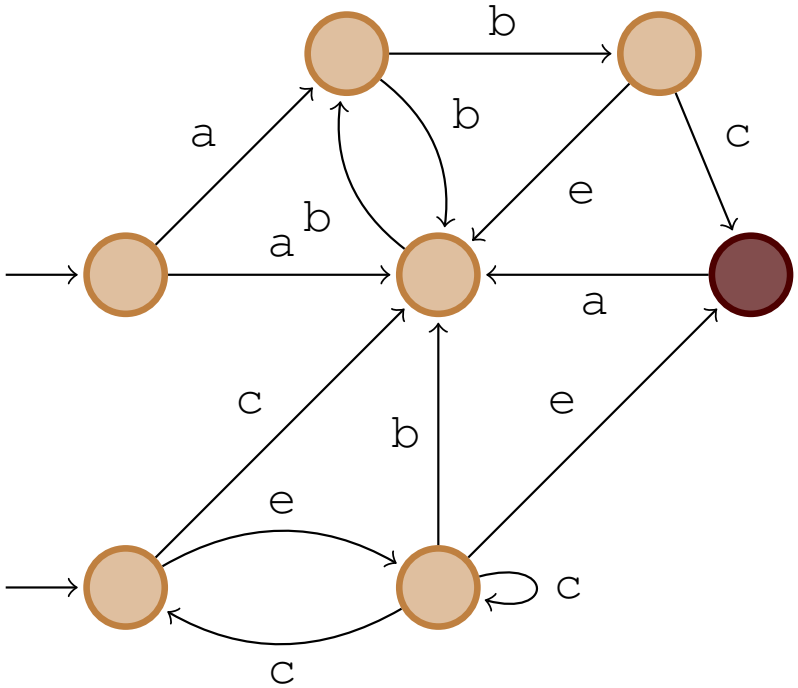
Consider the following finite-state system and assume that W consists of the single darker state.

$$G_W(Z) = \{x \in X \mid x \in W \text{ or } \exists u \in U(x) \quad \varnothing \neq \text{Post}_u(x) \subseteq Z\}$$



Consider the following finite-state system and assume that W consists of the single darker state.

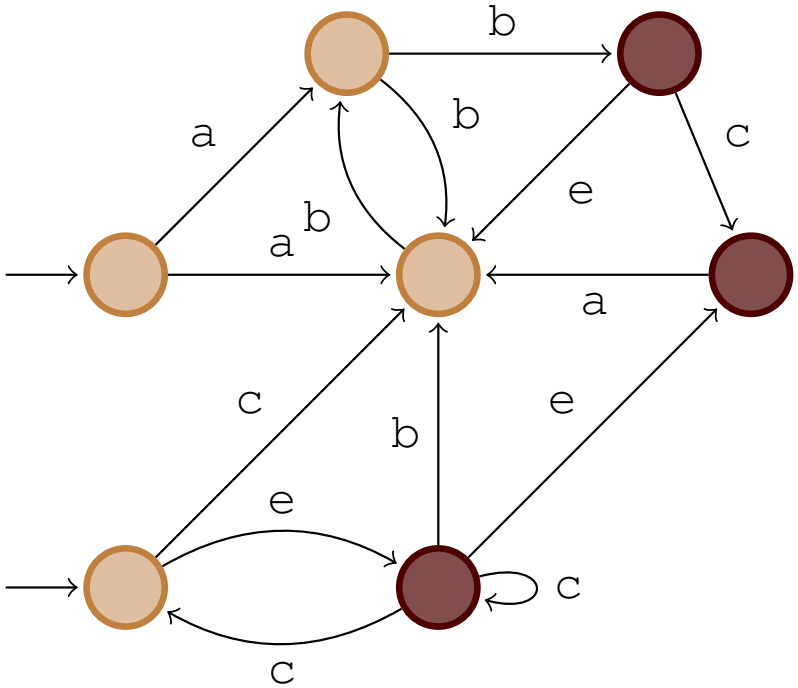
$$G_W(Z) = \{x \in X \mid x \in W \text{ or } \exists u \in U(x) \quad \varnothing \neq \text{Post}_u(x) \subseteq Z\}$$



$G_W(\varnothing)$

Consider the following finite-state system and assume that W consists of the single darker state.

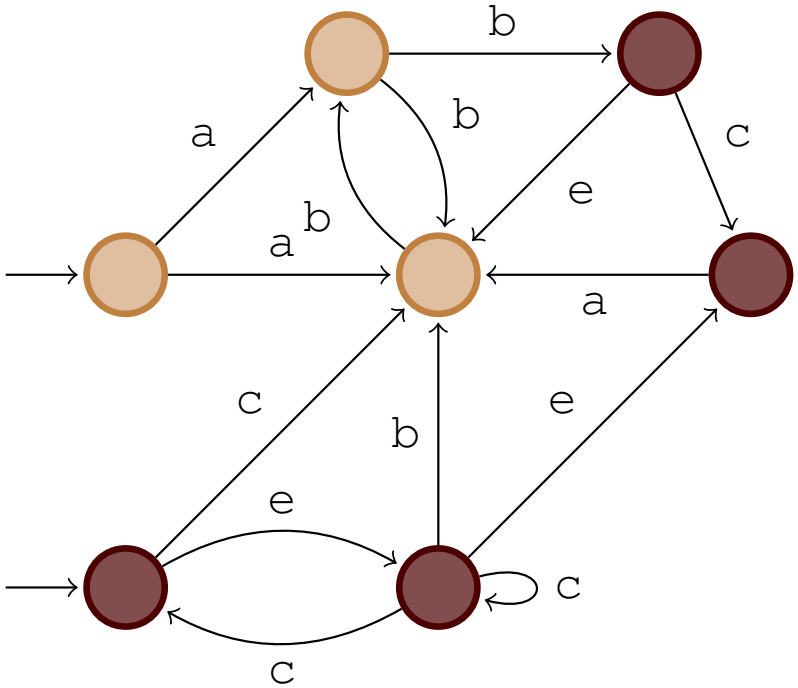
$$G_W(Z) = \{x \in X \mid x \in W \text{ or } \exists u \in U(x) \quad \varnothing \neq \text{Post}_u(x) \subseteq Z\}$$



$G_W^2(\varnothing)$

Consider the following finite-state system and assume that W consists of the single darker state.

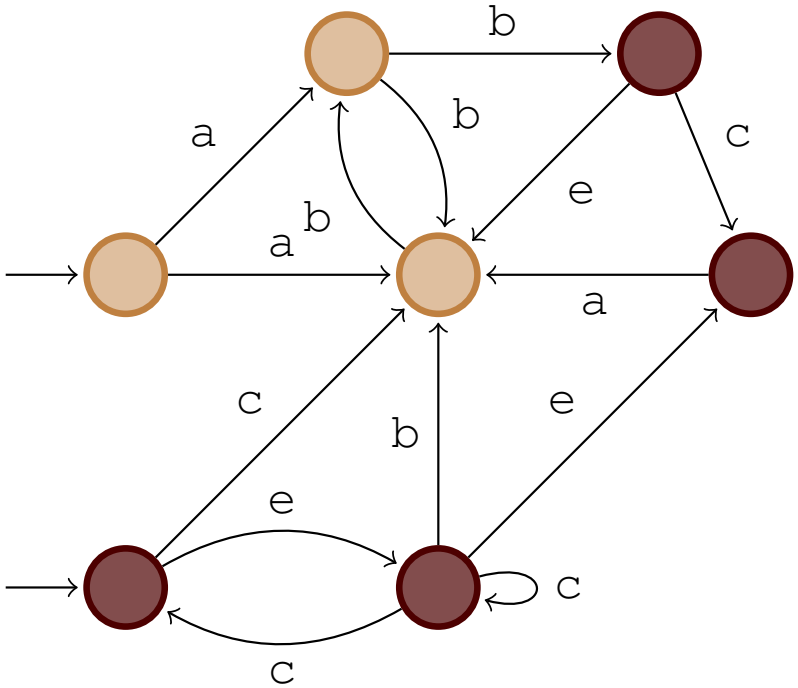
$$G_W(Z) = \{x \in X \mid x \in W \text{ or } \exists u \in U(x) \quad \varnothing \neq \text{Post}_u(x) \subseteq Z\}$$



$G_W^3(\varnothing)$

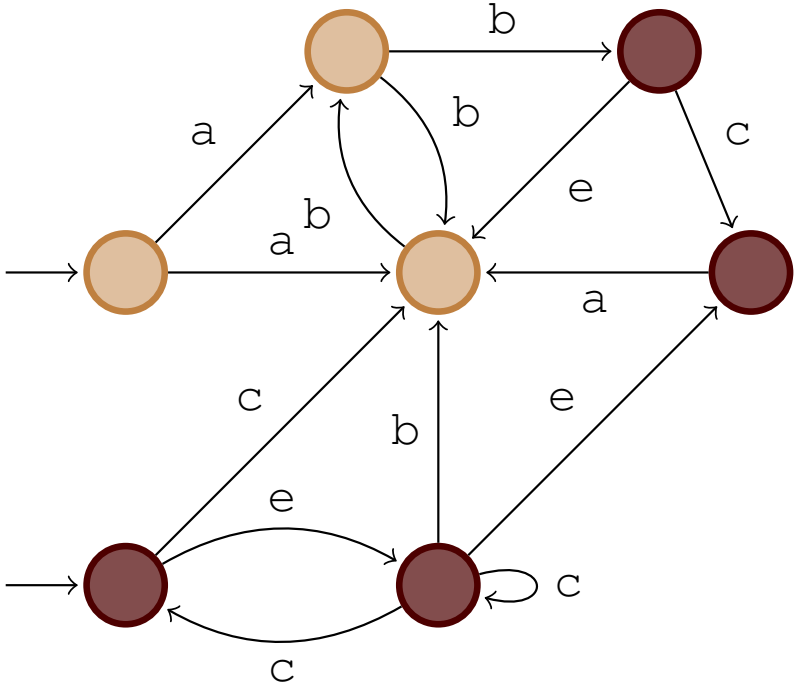
Consider the following finite-state system and assume that W consists of the single darker state.

$$G_W(Z) = \{x \in X \mid x \in W \text{ or } \exists u \in U(x) \quad \varnothing \neq \mathrm{Post}_u(x) \subseteq Z\}$$

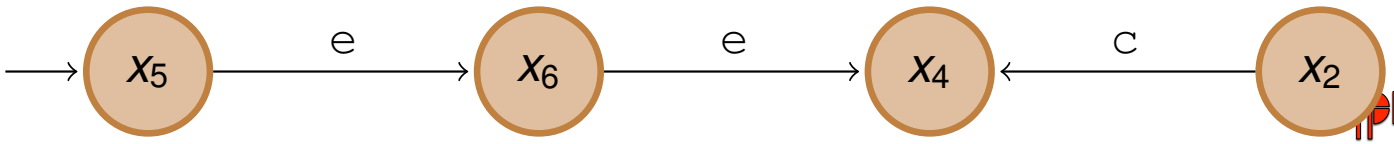


$G_W^4(\varnothing)$

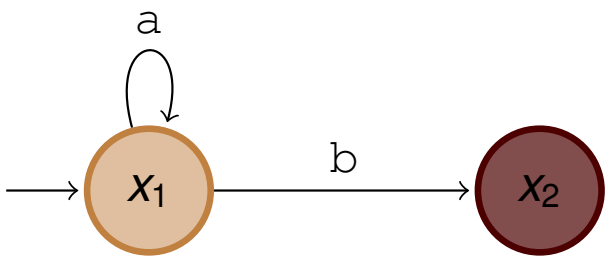
Consider the following finite-state system and assume that W consists of the single darker state.



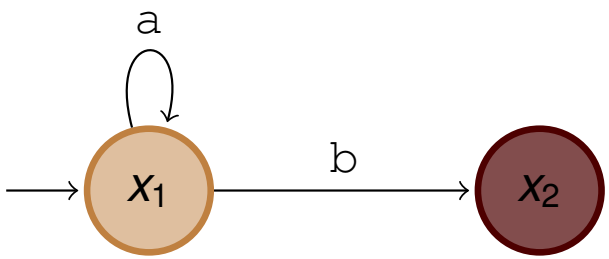
Controller:



An example to illustrate the non existence of maximal controllers:

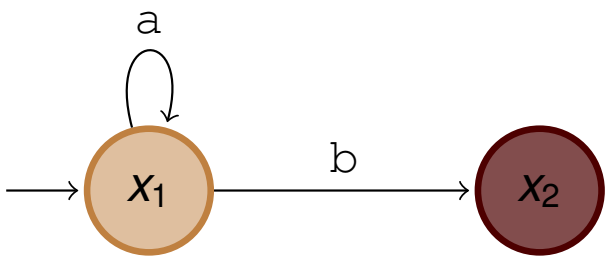


An example to illustrate the non existence of maximal controllers:



Any controller solving the reachability game necessarily loops $k \geq 0$ times on the state x_1 before reaching the state x_2.

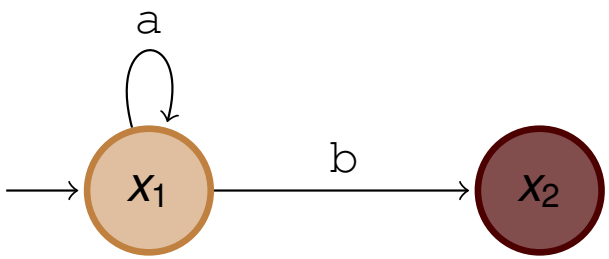
An example to illustrate the non existence of maximal controllers:



Any controller solving the reachability game necessarily loops $k \geq 0$ times on the state x_1 before reaching the state x_2.

However, we can always construct a less restrictive controller that loops $k + 1$ times on the state x_1 before reaching the state x_2!

An example to illustrate the non existence of maximal controllers:



Any controller solving the reachability game necessarily loops $k \geq 0$ times on the state x_1 before reaching the state x_2.

However, we can always construct a less restrictive controller that loops $k+1$ times on the state x_1 before reaching the state x_2!

If we have the possibility to set the initial state, then we can relax $X_0 \subseteq Z$ to $Z \cap X_0 \neq \varnothing$ so that there exists at least one initial state belonging to Z.

The algorithms for safety and reachabiltiy games both have polynomial complexity on the number of states of the system being controlled.

The algorithms for safety and reachabiltiy games both have polynomial complexity on the number of states of the system being controlled.

The current implementation in PESSOA uses Binary Decision Diagrams to encode systems up to one million states and can solve safety and reachability games in minutes.

The algorithms for safety and reachabiltiy games both have polynomial complexity on the number of states of the system being controlled.

The current implementation in PESSOA uses Binary Decision Diagrams to encode systems up to one million states and can solve safety and reachability games in minutes.

Although these techniques cannot be directly used on infinite-state systems, they can be used when symbolic abstractions exist.