Discrete and Continuous Dynamical Systems

Attila Magyar

University if Pannonia
Faculty of Information Technology
Department of Electrical Engineering and Information Systems

Discrete and continuous dynamical systems:

Analysis of discrete event systems

April 18, 2018

Overview

Nondeterminism

Possible sources of nonterminism

- Stochastic transitions (model is not detailed enough)
- Unobservable events

Problem: The actual state of the automaton is unknown by knowing the sequence of observable events

Nondeterministic Automata

Definition (Nondeterministic automaton)

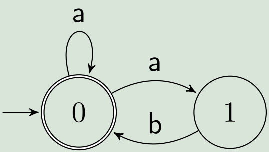
A nondeterministic automaton G_{nd} is a quintuple

$$G_{nd} = (X, E \cup \{\varepsilon\}, f_{nd}, x_0, X_m)$$

where all the objects have the same interpretation as in the definition of deterministic automaton except

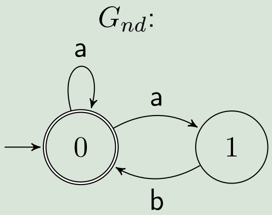
1. f_{nd} is a function $f_{nd} : X \times E \cup \{\varepsilon\} \to 2^X$, i.e. $f_{nd}(x, e) \subseteq X$ whenever it is defined.
2. The initial state may itself be a set of states, $x_0 \subseteq X$

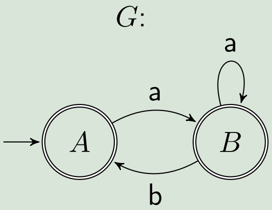
Example (A simple nondeterministic automaton)



Motivating example

Example (Nondeterministic and deterministic automata)





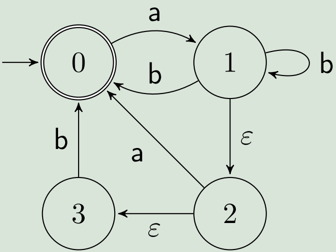
- State A corresponds to 0
- State B corresponds to $\{0, 1\}$

1. $f(A, a) = B$ and $f_{nd}(0, a) = \{0, 1\}$
2. $f(A, b)$ and $f_{nd}(0, b)$ are undefined
3. $f(B, a) = B$ and $f_{nd}(0, a) = \{0, 1\}$ ($f_{nd}(1, a)$ undefined)
4. $f(B, b) = A$ and $f_{nd}(1, b) = \{0\}$ ($f_{nd}(0, b)$ undefined)

The automata G_{nd} and G are language-equivalent! G is called an observer of G_{nd}

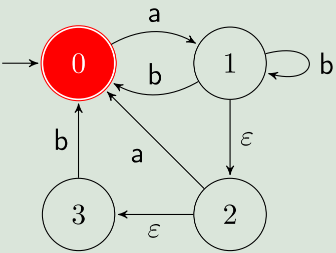
Another example

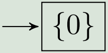
Example (Constructive example)



Another example

Example (Constructive example)

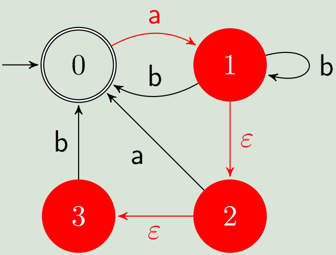


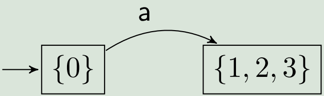


From state 0 we cannot get to other states reading ε

Another example

Example (Constructive example)

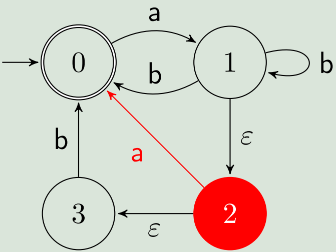


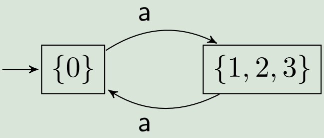


From state 0 we can get to states $1, 2$ and 3 reading a (or $a\varepsilon$ or $a\varepsilon\varepsilon$)

Another example

Example (Constructive example)

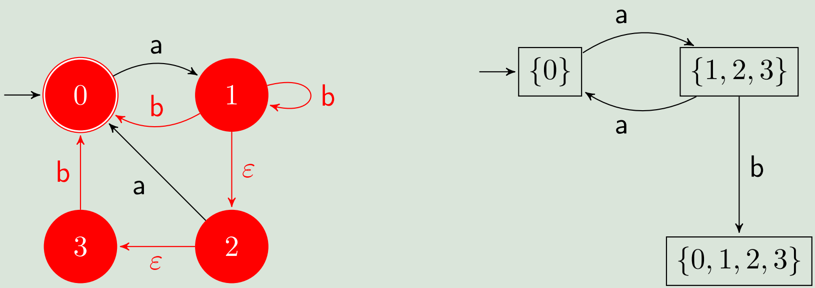




From the set of states $\{1, 2, 3\}$ we can get to state 0 reading a (from 2)

Another example

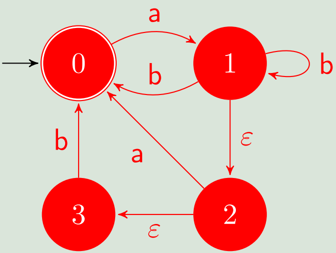
Example (Constructive example)

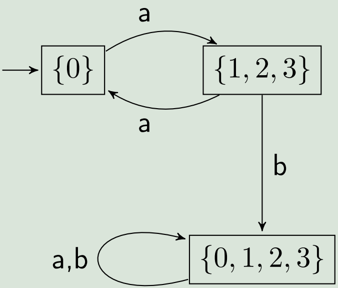


Reading b in any of the states $1, 2, 3$ leads to the set of states $\{0, 1, 2, 3\}$

Another example

Example (Constructive example)

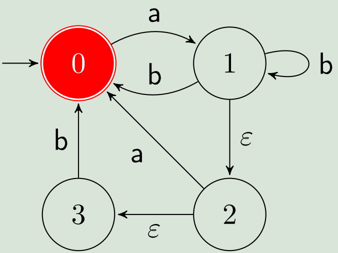


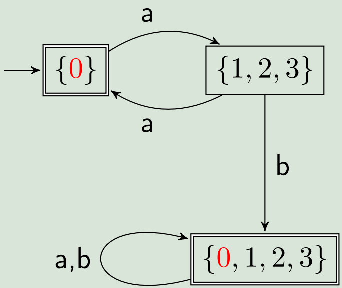


Reading a or b in any of the states $0, 1, 2, 3$ leads to the same set

Another example

Example (Constructive example)





Accepting states are the ones that contain 0

Reachability function

ε-reachability function

$$\varepsilon R(x) = \{p \in X : p \text{ is reachable from } x \text{ by } \varepsilon\}$$
$$\varepsilon R(B) = \cup_{x \in B} \varepsilon R(x)$$

Extended transition mapping

$$f_{nd}^{ext}(x, \varepsilon) = \varepsilon R(x)$$
$$f_{nd}^{ext}(x, ue) = \varepsilon R[\{z : z \in f_{nd}(y, e) \text{ for some state } y \in f_{nd}^{ext}(x, u)\}]$$

Observer automata

Procedure of building an observer $Obs(G_{nd})$

Step 1: Define $x_{0,obs} = \varepsilon R(x_0)$. Set $X_{obs} = \{x_{0,obs}\}$.

Step 2: for each $B \in X_{obs}$ and $e \in E$

$$f_{obs}(B, e) = \varepsilon R(\{x \in X : (\exists x_e \in B)\ [x \in f_{nd}(x_e, e)]\})$$

Step 3: Repeat Step 2 until the accessible part of $Obs(G_{nd})$ has been constructed

Step 4: $X_{m,obs} = \{B \in X_{obs} : B \cup X_m \neq \emptyset\}$

Important properties

- $Obs(G_{nd})$ is a deterministic automaton
- $\mathcal{L}(Obs(G_{nd})) = \mathcal{L}(G_{nd})$
- $\mathcal{L}_m(Obs(G_{nd})) = \mathcal{L}_m(G_{nd})$

Important in studying partially observed DES

Observer automata

Main idea

An outside observer that knows the system model G_{nd} but only observes the transitions labelled by the events in E will start with $x_{0,obs}$ as its estimate of the state G_{nd}. Upon observing event $e \in E$, this outside observer will update its state estimate to $f_{obs}(x_{0,obs}, e)$ as this set represents all the states where G_{nd} could be after executing the string e preceded /followed by ε.

Partially observed DES

- ε-transitions were defined to describe unobservable events
- Let us define genuine events for this phenomenon: unobservable events $E = E_{uo} \cup E_o$ where $E_{uo} \cap E_O = \emptyset$
- Instead of NFA, DFA might be used with unobservable events
- Treat unobservable events as they were ε

Definition (Unobservable reach)

The unobservable reach of state $x \in X$ denoted by $UR(x)$ is

$$UR(x) = \{y \in X : \ (\exists t \in E_{uo}^*)[f(x,t) = y]\}$$

The definition can be extended to sets of states $B \subseteq X$ by

$$UR(B) = \cup_{x \in B} UR(x)$$

Observer for automaton G with unobservable events

Let $G = (X, E, f, x_0, X_m)$ be a deterministic automaton and let $E = E_{uo} \cup E_o$. Then $Obs(G) = (X_{obs}, E_o, f_{obs}, x_{0,obs}, X_{m,obs})$ can be built as follows

Step 1: Define $x_{0,obs} = UR(x_0)$
set $X_{m,obs} = \{x_{0,obs}\}$

Step 2: For each $B \in X_{obs}$ and $e \in E_o$ define

$$f_{obs}(B, e) = UR(\{x \in X : (\exists x_e \in B)[x \in f(x_e, e)]\})$$

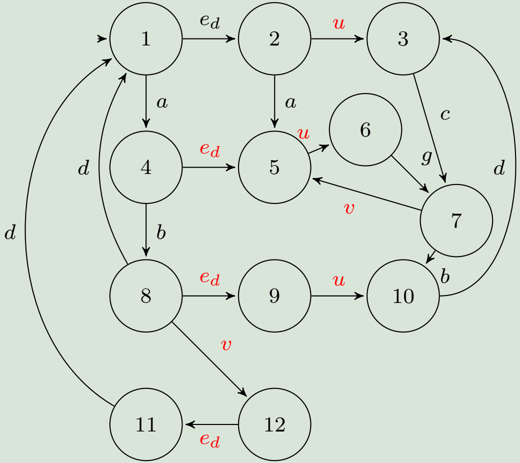
whenever $f(x_e, e)$ is defined for some $x_e \in B$

Step 3: Repeat Step 2 until the entire accessible part of $Obs(G)$ has been constrcted

Step 4: $X_{m,obs} = \{B \in X_{obs} : \; B \cap X_m \neq \emptyset\}$

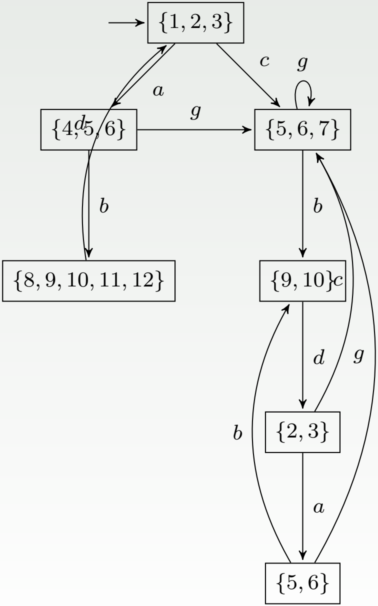
Observer with unobservable events

Example (Automaton with unobservable events)



- Set of unobservable events

$$E_{uo} = \{e_d, u, v\}$$


$\{1,2,3\}$
c
g
a
d
$\{4,5,6\}$
g
$\{5,6,7\}$
b
b
$\{8,9,10,11,12\}$
$\{9,10\}$
c
d
g
b
$\{2,3\}$
a
$\{5,6\}$

Overview

Diagnosing events

Event diagnosis: determining if certain unobservable event could have been occured or must have occured in the string of events executed by the system

Definition (Diagnosability)

Unobservable event e_d is not diagnosable in language $\mathcal{L}\{G\}$ if there exist two strings s_N and s_Y in $\mathcal{L}\{G\}$ that satisfy the following conditions:

- s_Y contains e_d and s_N does not
- s_Y is of arbitrary long length after e_d
- $P(s_N) = P(s_Y)$

when no such pair of strings exists, e_d is said to be diagnosable in $\mathcal{L}\{G\}$

Diagnoser Automata

- Diagnozers are similar to observers with the difference that labels are attached to the states of G of $Diag(G)$:

  N No, e_d has not occured yet

  Y Yes,e_d has occured
- They are used to track the system behavior and diagnose, if possible, the prior occurrence of certain unobservable events
- If multiple events to be diagnosed, we can either build one diagnoser for each events to be diagnosed, or build a single diagnoser for all

Building $Diag(G)$

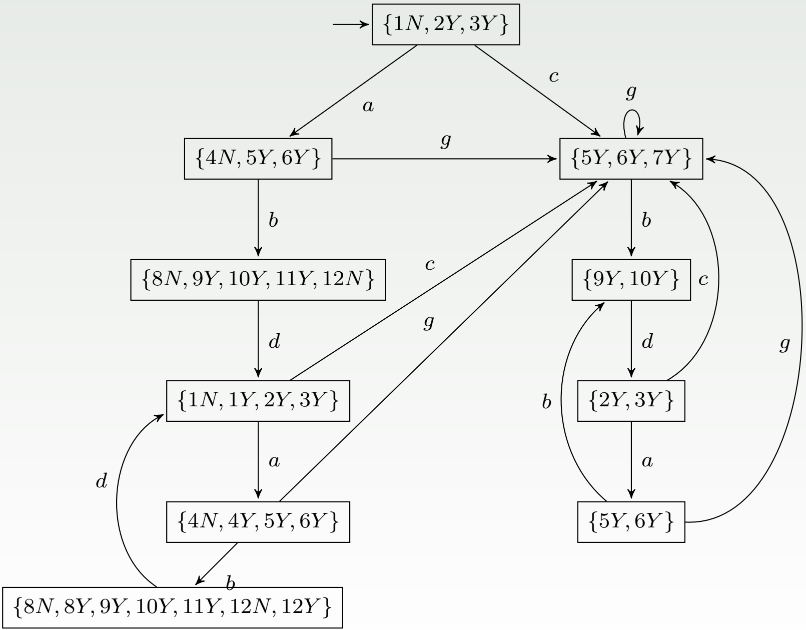
Mod. 1 When building the unobservable reach of x_0 of G:

1. Attach the label N to states that can be reached from x_0 by unobservable strings in $[E_{uo} \setminus \{e_d\}]^*$
2. Attach the label Y to states that can be reached from x_0 by unobservable strings that contain at least one occurrence of e_d
3. If state z can be reached both with and without executing e_d, then create two entries in the initial state set of $Diag(G)$: z_n and z_Y

Mod. 2 When building subsequent reachable states of $Diag(G)$:

1. Follow the rules for the transition function of $Obs(G)$, but with the above modified way to build unobservable reaches with state labels
2. Propagate the label Y to indicate that e_d has occurred in the process of reaching z and thus in tha process of reaching the new state

Mod. 3 No set of marked states is defined for $Diag(G)$


{1N, 2Y, 3Y}
a
c
g
{4N, 5Y, 6Y}
g
{5Y, 6Y, 7Y}
b
b
{8N, 9Y, 10Y, 11Y, 12N}
c
{9Y, 10Y}
c
g
d
d
g
{1N, 1Y, 2Y, 3Y}
b
{2Y, 3Y}
a
a
d
{4N, 4Y, 5Y, 6Y}
{5Y, 6Y}
b
{8N, 8Y, 9Y, 10Y, 11Y, 12N, 12Y}