

System Modelling

Introduction

Finite State Machines

Petri Nets

Untimed Model of Computation

Synchronous Model of Computation

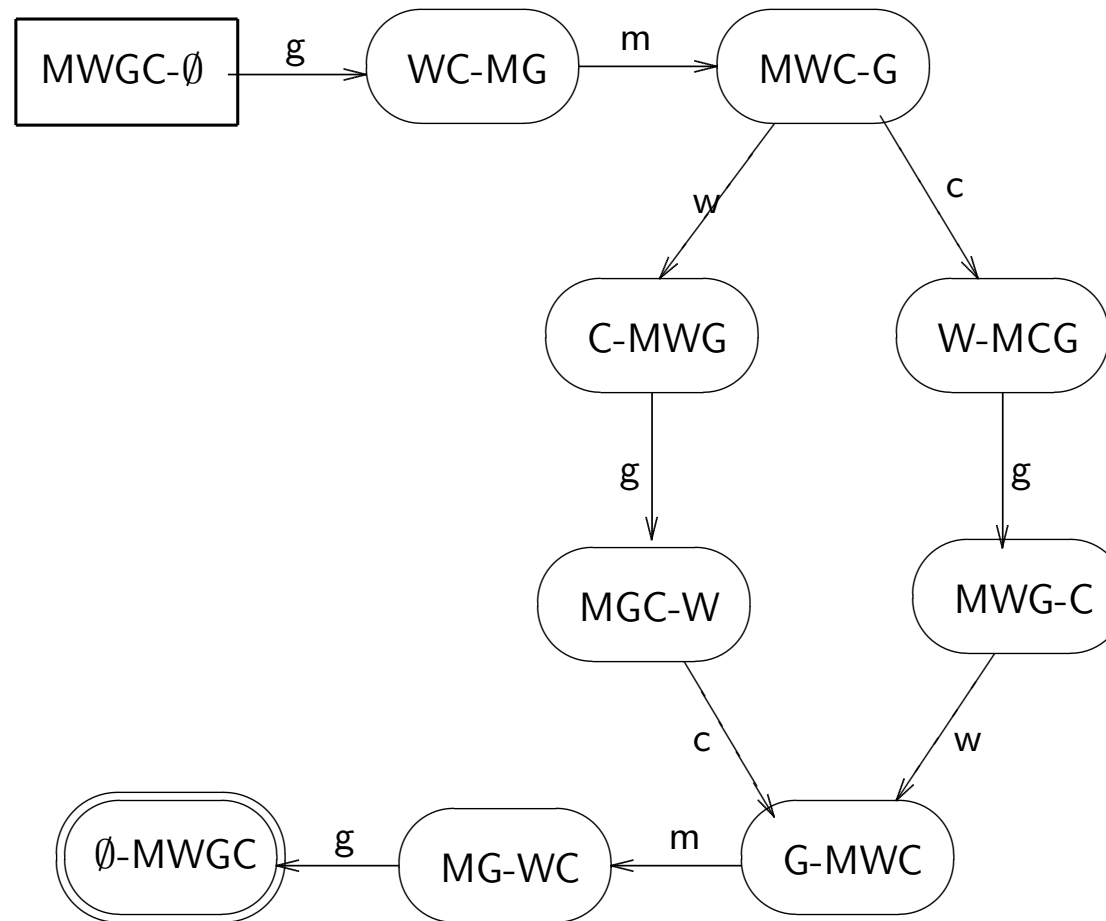
Timed Model of Computation

Integration of Computational Models

Tightly Coupled Process Networks



The Man, Wolf, Goat, Cabbage Problem



Finite State Machine Definition

Definition: A **finite state machine** is a five-tuple

$$(\Sigma, X, g, x_0, F)$$

where

Σ is a finite **alphabet**

X is a finite **set of states**

g is a **state transition function**, $g : X \times \Sigma \rightarrow X$

x_0 is the **initial state**, $x_0 \in X$

F is the **set of final states**, $F \subseteq X$.

The Man-Wolf-Goat-Cabbage Problem as FSM

$M = (\Sigma, X, g, x_0, F)$, where

$$\Sigma = \{m, w, g, c\}$$

$$X = \{(x, y) \mid x, y \subseteq \{M, W, G, C\} \text{ and } y = \{M, W, G, C\} \setminus x\}$$

$$x_0 = \{(\{M, W, G, C\}, \emptyset)\}$$

$$F = \{(\emptyset, \{M, W, G, C\})\}$$

$$g((\{M, W, G, C\}, \emptyset), g) = (\{W, C\}, \{M, G\}),$$

$$g((\{W, C\}, \{M, G\}), m) = (\{W, C, M\}, \{G\}),$$

$$g((\{M, W, C\}, \{G\}), w) = (\{C\}, \{M, W, G\}),$$

$$g((\{M, W, C\}, \{G\}), c) = (\{W\}, \{M, G, C\}),$$

$$g((\{C\}, \{M, W, G\}), g) = (\{M, G, C\}, \{W\}),$$

$$g((\{W\}, \{M, G, C\}), g) = (\{M, W, G\}, \{C\}),$$

$$g((\{M, G, C\}, \{W\}), c) = (\{G\}, \{M, W, C\}),$$

$$g((\{M, W, C\}, \{G\}), w) = (\{G\}, \{M, W, C\}),$$

$$g((\{G\}, \{M, W, C\}), m) = (\{M, G\}, \{W, C\}),$$

$$g((\{M, G\}, \{W, C\}), g) = (\emptyset, \{M, W, G, C\}).$$



Some Definitions

- Σ^* is the set of strings formed from elements of alphabet Σ .
- Let $g : X \times \Sigma \rightarrow X$ be a state transition function. Then $g^* : X \times \Sigma^* \rightarrow X$ is defined as follows $\forall x \in X, a \in \Sigma, r \in \Sigma^*$.

$$g^*(x, \epsilon) = x$$

$$g^*(x, "a") = g(x, a)$$

$$g^*(x, "a" + r) = g^*(g(x, a), r)$$

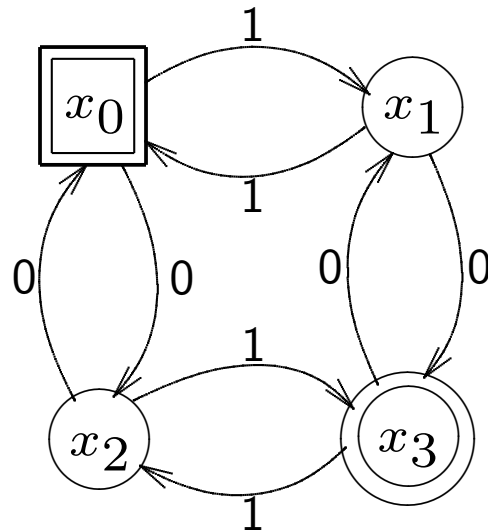
More Definitions

- A finite state machine $M = (\Sigma, X, g, x_0, F)$ **accepts** an input string $r \in \Sigma^*$ iff $g^*(x_0, r) \in F$.
- The **language** accepted by M , designated $L(M)$, is the set of strings accepted by M , i.e.

$$L(M) = \{r \in \Sigma^* \mid g^*(x_0, r) \in F\}.$$

E.g. in the Man-Wolf-Goat-Cabbage problem the machine M accepts strings “gmwgcmg” and “gmcgwmg” but no other strings. Thus $L(M) = \{\text{“gmwgcmg”}, \text{“gmcgwmg”}\}$.

FSM Accepting an Even Number of Symbols



$M = (\Sigma, X, g, x_0, F)$, where

$$\Sigma = \{0, 1\}$$

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

$$F = \{x_0, x_3\}$$

$$\begin{array}{ll} g(x_0, 0) = x_2, & g(x_0, 1) = x_1 \\ g(x_1, 0) = x_3, & g(x_1, 1) = x_0 \\ g(x_2, 0) = x_0, & g(x_2, 1) = x_3 \\ g(x_3, 0) = x_1, & g(x_3, 1) = x_2. \end{array}$$

- x_0 represents an even number of both zeros and ones;
- x_3 represents an odd number of both zeros and ones;
- x_1 represents an even number of zeros and an odd number of ones;
- x_2 represents an odd number of zeros and an even number of ones;
- $F = \{x_0, x_3\} \Rightarrow L(M) = \{r \in \Sigma^* \mid \#r \bmod 2 = 0\}$.

Non-finite State Machines

Definition: A **state machine** is a five-tuple

$$(\Sigma, X, g, x_0, F)$$

where

- Σ is a **countable alphabet**
- X is a **countable set of states**
- g is a **state transition function**, $g : X \times \Sigma \rightarrow X$
- x_0 is the **initial state**, $x_0 \in X$
- F is the **set of final states**, $F \subseteq X$.

Nondeterministic Finite State Machines

Definition: A **nondeterministic finite state machine** is a five tuple

$$(\Sigma, X, g, x_0, F)$$

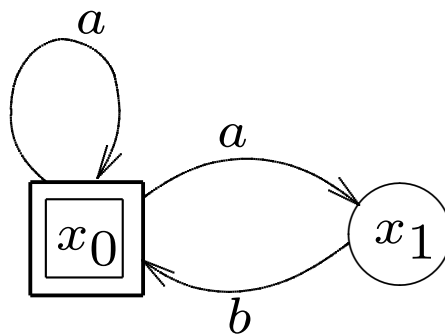
where

- Σ is a finite **alphabet**
- X is a finite **set of states**
- g is a **state transition mapping**, $g : X \times \Sigma \rightarrow \wp(X)$
- x_0 is the **initial state**, $x_0 \in X$
- F is the **set of final states**, $F \subseteq X$.



An Example Nondeterministic FSM

$M = (\Sigma, X, x_0, g, F)$ is defined as follows.



$$\Sigma = \{a, b\}$$

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

$$F = \{x_0\}$$

$$\begin{aligned} g(x_0, a) &= \{x_0, x_1\}, & g(x_0, b) &= \emptyset \\ g(x_1, a) &= \emptyset, & g(x_1, b) &= \{x_0\} \end{aligned}$$

Extended Transition Function for Nondeterministic FSM

$g : X \times \Sigma \rightarrow \wp(X)$ is a state transition function

Then $g^* : X \times \Sigma^* \rightarrow \wp(X)$ is defined as follows $\forall x \in X, a \in \Sigma, r \in \Sigma^*$.

$$g^*(x, \epsilon) = \{x\}$$

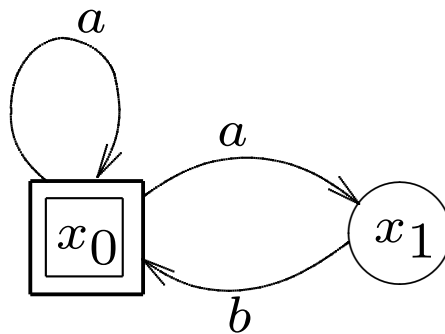
$$g^*(x, a) = g(x, a)$$

$$g^*(x, "a" + r) = \{z \mid z \in g^*(y, r) \text{ for some } y \in g(x, "a")\}$$



An Execution Example of a Nondeterministic FSM

For input string "abaab":



$$\begin{aligned} &g^*({x_0}, \text{"abaab"}) && \text{step 1} \\ &= g^*({x_0, x_1}, \text{"baab"}) && \text{step 2} \\ &= g^*({x_0}, \text{"aab"}) && \text{step 3} \\ &= g^*({x_0, x_1}, \text{"ab"}) && \text{step 4} \\ &= g^*({x_0, x_1}, \text{"b"}) && \text{step 5} \\ &= g^*({x_0}, \epsilon) \\ &= {x_0} \end{aligned}$$

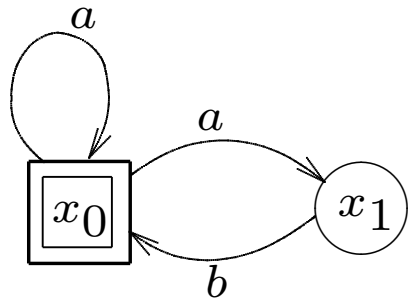
Equivalence of Deterministic and Nondeterministic FSMs

Theorem: For every deterministic finite state machine M there is a nondeterministic finite state machine M_{ND} with $L(M) = L(M_{ND})$.

Theorem: For every nondeterministic finite state machine M there is a deterministic finite state machine M_D with $L(M) = L(M_D)$.



Construction of a Deterministic FSM



$$\begin{aligned}
 M &= (\Sigma, X, x_0, g, F) \\
 \Sigma &= \{a, b\} \\
 X &= \{x_0, x_1\} \\
 F &= \{x_0\}
 \end{aligned}$$

$$\begin{aligned}
 g(x_0, a) &= \{x_0, x_1\} \\
 g(x_0, b) &= \emptyset \\
 g(x_1, a) &= \emptyset \\
 g(x_1, b) &= \{x_0\}
 \end{aligned}$$

Deterministic machine:

$$\begin{aligned}
 M^D &= (\Sigma^D, X^D, x_0^D, g^D, F^D) \\
 \Sigma^D &= \Sigma \\
 X^D &= \{y_\emptyset, y_0, y_1, y_{01}\} \\
 x_0^D &= y_0 \\
 F^D &= \{y_0, y_{01}\}
 \end{aligned}$$

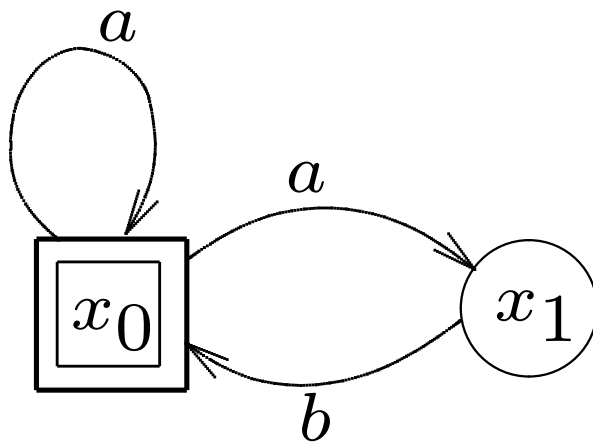
$$\begin{aligned}
 g^D(y_\emptyset, a) &= y_\emptyset, & g^D(y_\emptyset, b) &= y_\emptyset \\
 g^D(y_0, a) &= y_{01}, & g^D(y_0, b) &= y_0 \\
 g^D(y_1, a) &= y_\emptyset, & g^D(y_1, b) &= y_0 \\
 g^D(y_{01}, a) &= y_{01}, & g^D(y_{01}, b) &= y_0
 \end{aligned}$$

Intuitively, we have the following correspondence:

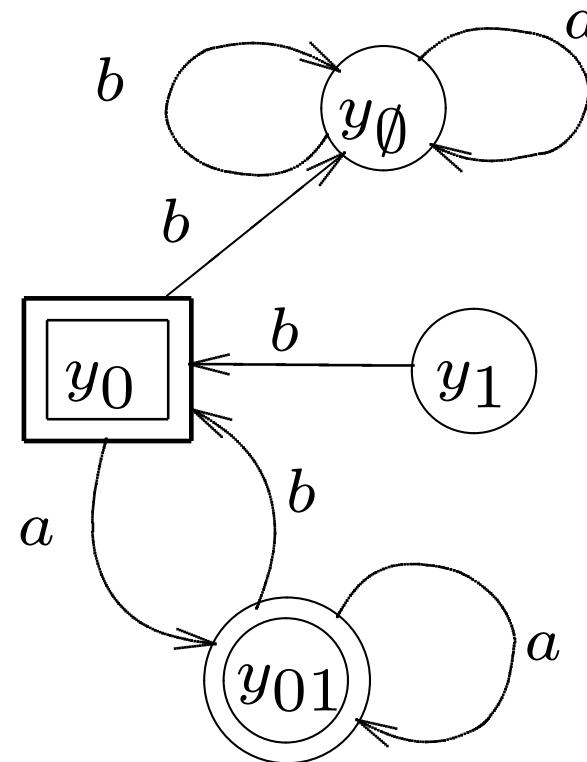
$$\begin{aligned}
 y_\emptyset &: \emptyset & : \text{"no possible state of } M\text{"} \\
 y_0 &: \{x_0\} & : x_0 \\
 y_1 &: \{x_1\} & : x_1 \\
 y_{01} &: \{x_0, x_1\} & : \text{"either state } x_0 \text{ or state } x_1\text{"} .
 \end{aligned}$$

Construction of a Deterministic FSM - cont'd

Nondeterministic FSM:



Deterministic FSM:



Finite State Machines with ϵ -Moves

Definition: A **nondeterministic finite state machine with ϵ -moves** is a five tuple

$$(\Sigma, X, g, x_0, F)$$

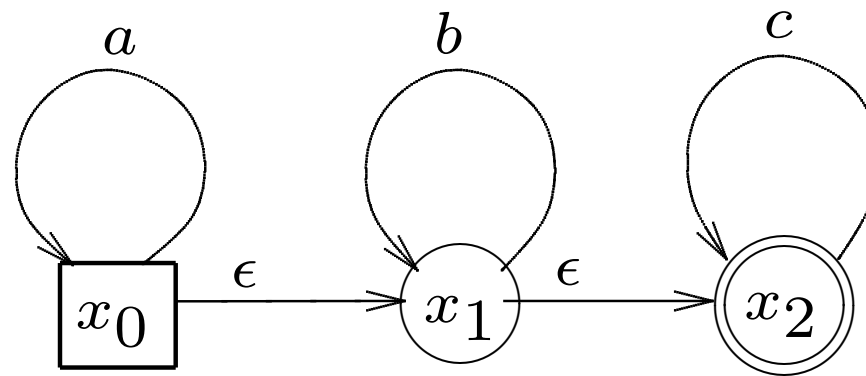
where

- Σ is a finite **alphabet**
- X is a finite **set of states**
- g is a **state transition mapping**, $g : X \times (\Sigma \cup \{\epsilon\}) \rightarrow \wp(X)$
- x_0 is the **initial state**, $x_0 \in X$
- F is the **set of final states**, $F \subseteq X$.



Finite State Machines with ϵ -Moves - cont'd

Example:



Theorem: For every nondeterministic finite state machine with ϵ -moves M there is a nondeterministic finite state machine M_D with $L(M) = L(M_D)$.

State Equivalence

Definition: Let $M = (\Sigma, X, g, x_0, F)$ be a finite state machine. Two states x and y are **equivalent**, denoted by $x \sim y$, iff

$$\forall r \in \Sigma^* : g^*(x, r) \in F \Leftrightarrow g^*(y, r) \in F.$$

A set $R \subseteq X$ is a set of equivalent states iff

$$\forall x, y \in R, r \in \Sigma^* : x \sim y.$$



State Equivalence - cont'd

- (i) Two states $x \in F$ and $y \notin F$ are not equivalent because $g(x, \epsilon) \in F$ but $g(y, \epsilon) \notin F$.
- (ii) Two states $x, y \in X$, of which either both are in F or neither is in F , are equivalent if $\forall e \in \Sigma : g(x, e) = g(y, e)$.
- (iii) Two states $x, y \in X$, of which either both are in F or neither is in F , are equivalent if for some $e \in \Sigma : g(x, e) = y$ and $g(y, e) = x$ and $\forall e' \in \Sigma, e' \neq e : g(x, e') = g(y, e')$. Thus, for event e the states x and y are simply interchanged and for all other events the successor states for x and y are identical.
- (iv) We can generalize previous observation. A set $R \subseteq X$, for which either $R \subseteq F$ or $R \cap F = \emptyset$, is a set of equivalent states if $\forall x, y \in R, e \in \Sigma : g(x, e) = z \notin R \Rightarrow g(y, e) = z$. Intuitively, if for an event e the set R is left, it is left for all states in R and it leads to the same state outside R .



State Aggregation

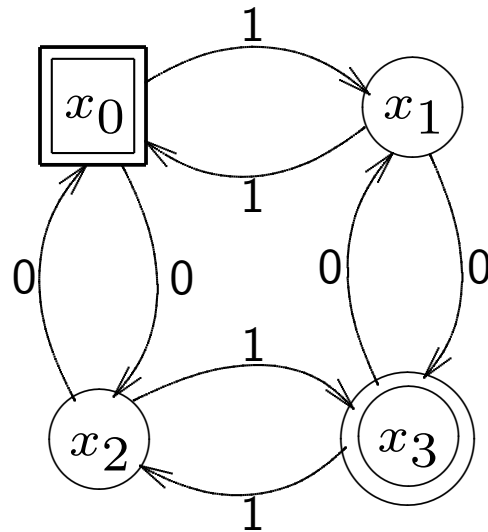
Algorithm:

- Step 1. for all pairs of states (x, y) set $L(x, y) := \emptyset$;
mark all pairs of states (x, y) with $x \in F$ and $y \notin F$;
- Step 2. for every pair (x, y) not marked
do set flag := 0;
- Step 2.1. for every $e \in \Sigma$;
do if $(g(x, e), g(y, e))$ is marked
then markall($L, (x, y)$);
set flag := 1;
- Step 2.2. if (flag = 0)
then for every $e \in \Sigma$
if $(g(x, e) \neq g(y, e))$
then add (x, y) to $L(g(x, e), g(y, e))$.

$$x \not\sim y \Rightarrow \text{forall } (x_1, y_1) \in L(x, y) : x_1 \not\sim y_1$$



State Aggregation Example - 1



Step 1:

	x_0	x_1	x_2	x_3
x_0	×	●	●	
x_1	×	×		●
x_2	×	×	×	●
x_3	×	×	×	×

(x_0, x_3) and (x_1, x_2) are not marked.

State Aggregation Example - 2

Step 2: We consider the pairs $\{(x_1, x_2), (x_0, x_3)\}$. First iteration through the loop of (step 2): We set $(x, y) := (x_1, x_2)$ and $\text{flag} := 0$.

Step 2.1: The list of events to consider is $\{0, 1\}$.

$e := 0$: $g(x_1, 0) = x_3$, $g(x_2, 0) = x_0$ and (x_0, x_3) are not marked.

$e := 1$: $g(x_1, 1) = x_0$, $g(x_2, 1) = x_3$ and (x_0, x_3) are not marked.

Step 2.2: Since flag is not set, we enter (step 2.2).

$e := 0$: $g(x_1, 0) = x_3 \neq g(x_2, 0) = x_0$;
we set $L(x_0, x_3) := \{(x_1, x_2)\}$.

$e := 1$: $g(x_1, 1) = x_0 \neq g(x_2, 1) = x_3$;
 $L(x_0, x_3) := L(x_0, x_3) \cup (x_1, x_2) = \{(x_1, x_2)\}$.

State Aggregation Example - 3

Step 2: $(x, y) := (x_0, x_3)$ and $\text{flag} := 0$.

Step 2.1: The list of events to consider is $\{0, 1\}$.

$e := 0$: $g(x_0, 0) = x_2$, $g(x_3, 0) = x_1$ and (x_1, x_2) is not marked.

$e := 1$: $g(x_0, 1) = x_1$, $g(x_3, 1) = x_2$ and (x_1, x_2) is not marked.

Step 2.2: Since flag has not been set, we enter (step 2.2). Again we have to go through the set of events.

$e := 0$: $g(x_0, 0) = x_2 \neq g(x_3, 0) = x_1$;

$L(x_1, x_2) := \{(x_0, x_3)\}$.

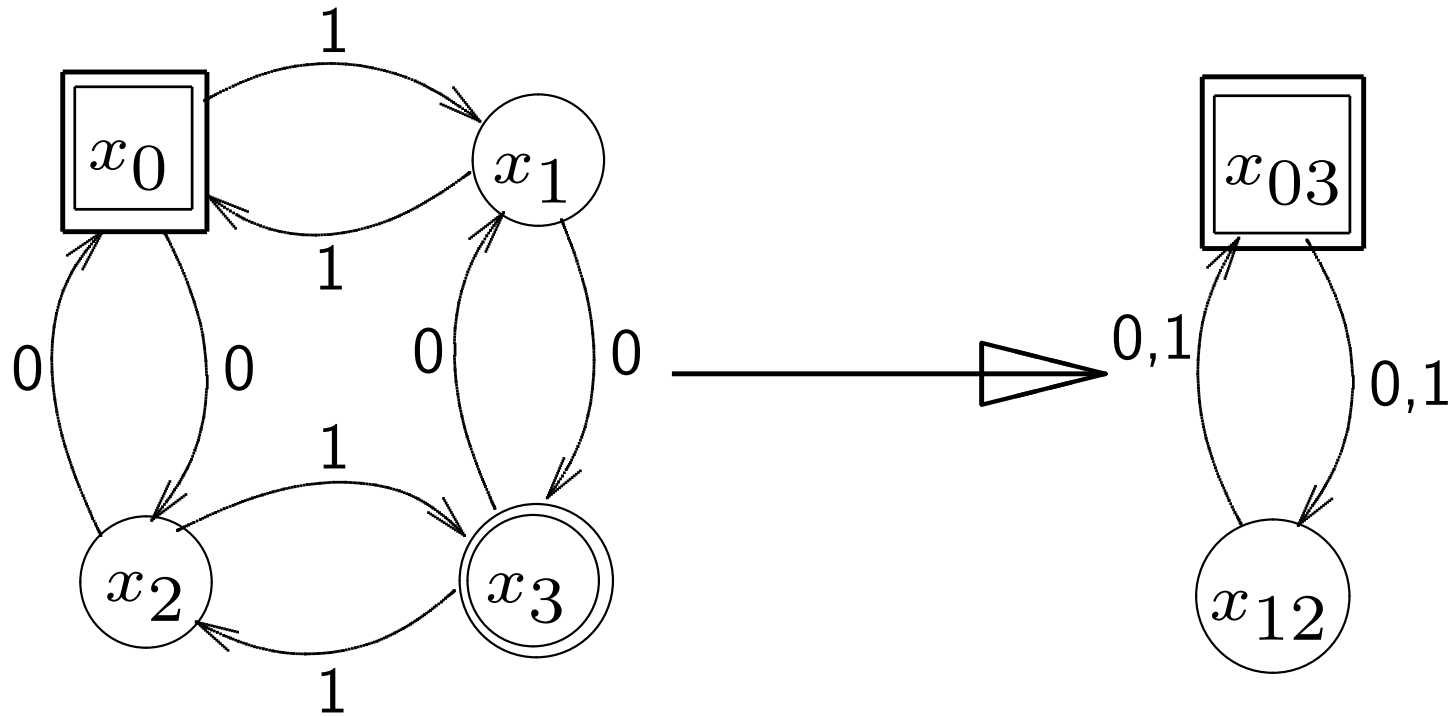
$e := 1$: $g(x_0, 1) = x_1 \neq g(x_3, 1) = x_2$;

$L(x_0, x_3) := L(x_1, x_2) \cup (x_0, x_3) = (x_0, x_3)$.

End: (x_0, x_3) and (x_1, x_2) are unmarked; $\Rightarrow x_0 \sim x_3$ and $x_1 \sim x_2$.



State Aggregation Example - 4



Operations on Languages

Definition: Σ is a set of symbols; L , L_1 , and L_2 are sets of strings from Σ^* . The **concatenation of two languages** L_1 and L_2 is

$$L_1L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$$

L^n is the concatenation of a language n times with itself, i.e.

$$L^0 = \{\epsilon\}, L^i = LL^{i-1}, \forall i > 0.$$

The **Kleene closure** of L is $L^* = \bigcup_{i=0}^{\infty} L^i$.

The **positive Kleene closure**, L^+ , is the set $L^+ = \bigcup_{i=1}^{\infty} L^i$.



Operations on Languages - Examples

Let $\Sigma = \{a, b, c\}$, $L_1 = \{\epsilon, a, abb\}$ and $L_2 = \{c\}$. Then

$$L_1^2 = \{\epsilon, a, abb, aa, aabb, abba, abbabb\}$$

$$L_2^3 = \{ccc\}$$

$$L_1L_2 = \{c, ac, abbc\}$$

$$L_1^* = \{\epsilon, a, abb, aa, abba, aabb, abbabb, aaa, abbaa, aabba, \dots\}$$

$$L_2^+ = \{c, cc, ccc, \dots\}$$



Regular Expressions and Regular Languages

Definition: Let Σ be an alphabet. The **regular expressions** and the sets that they denote are defined as follows.

1. \emptyset is a regular expression and denotes the empty set.
2. ϵ is a regular expression and denotes $\{\epsilon\}$.
3. For each $a \in \Sigma$, a is a regular expression and denotes $\{a\}$.
4. If r and s are regular expressions with $L(r) = R, L(s) = S$, then
 - (a) $(r + s)$ is a regular expression with $L(r + s) = R \cup S$;
 - (b) (rs) is a regular expression with $L(rs) = RS$;
 - (c) r^* is a regular expression with $L(r^*) = R^*$.

Definition: Every set (language) that can be denoted by a regular expression is a **regular set** (**regular language**).



Regular Expression Examples

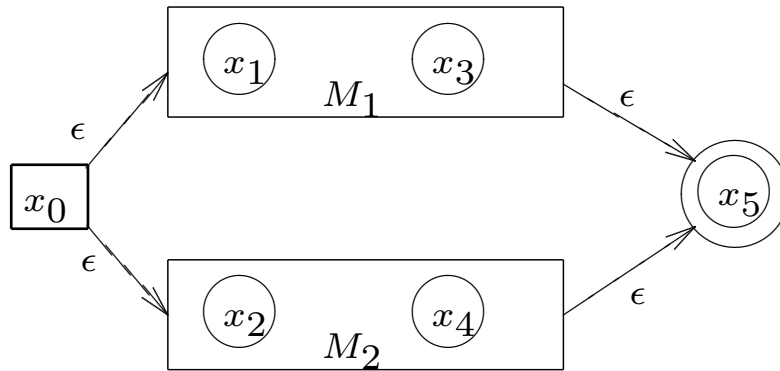
- aa : $L(aa) = \{aa\}$
- a^* : $L(a^*) = \{\epsilon, a, aa, \dots\}$
- $(a + ab)^*$: $L((a + ab)^*) =$ all strings starting with a and with at most 1 consecutive b ;
- $aab(a + b)^*baa$: $L(aab(a + b)^*baa) =$ all strings of a 's and b 's beginning with aab and ending with baa ;

Regular Expressions and Finite State Machines

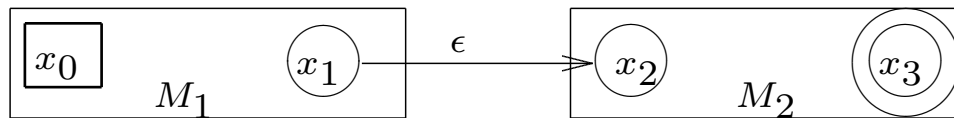
Theorem: A language can be denoted by a regular expression iff there exists a nondeterministic finite state machine with ϵ -transitions which accepts the language.



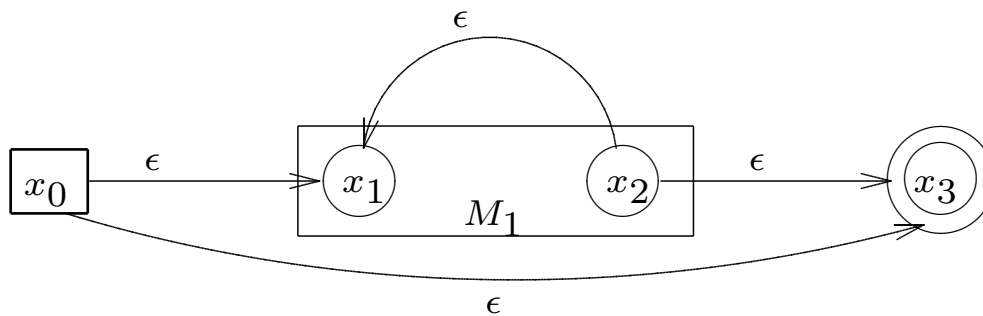
FSM Templates for Regular Expressions



Union: $r + s$



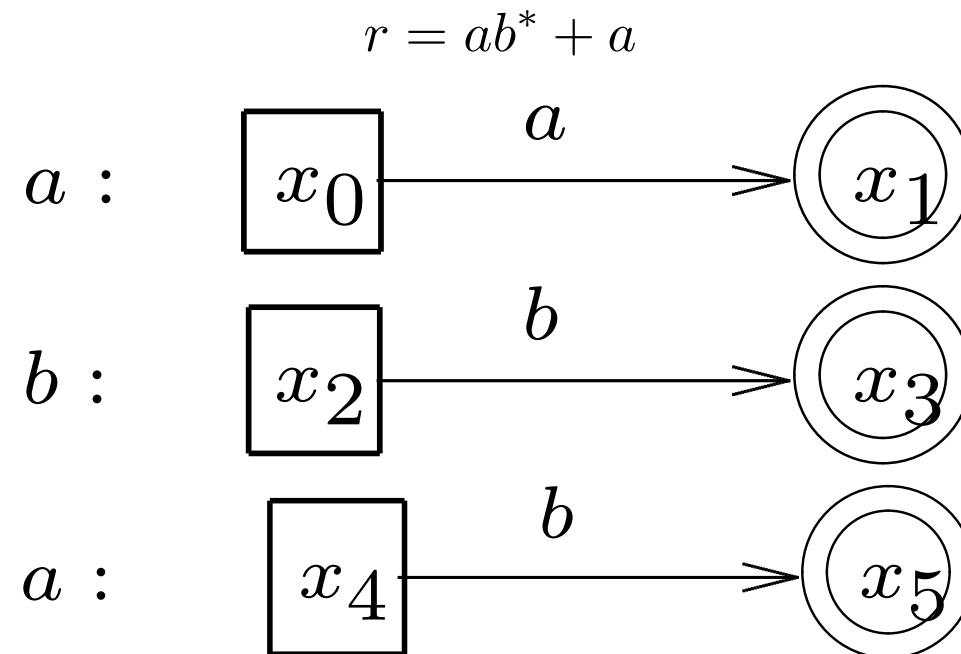
Concatenation: rs



Kleene Closure: r^*

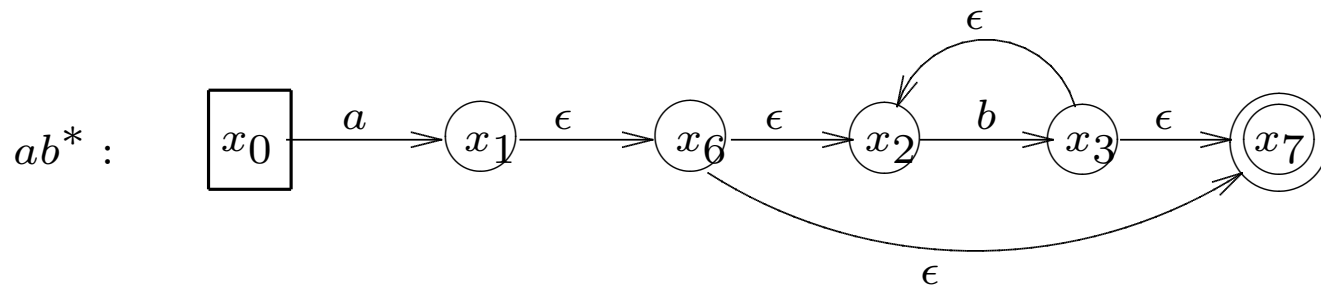
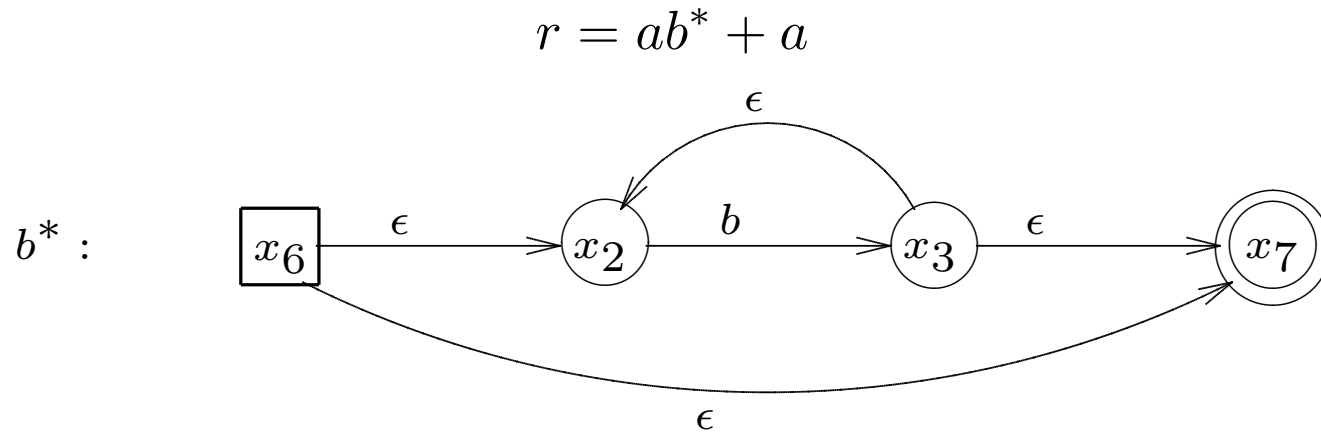


FSM Construction Example - 1



FSM Construction Example - 2

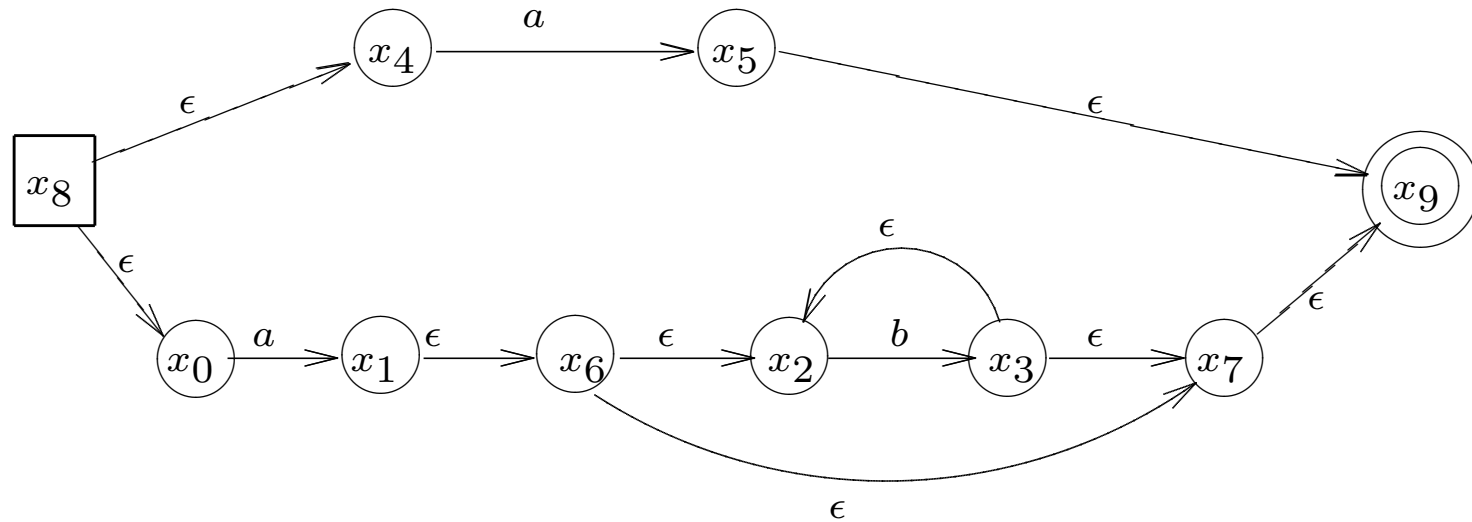
$$r = ab^* + a$$



FSM Construction Example - 3

$$r = ab^* + a$$

$ab^* + b :$



Moore Finite State Machines

Definition: A deterministic **Moore machine** is a six-tuple

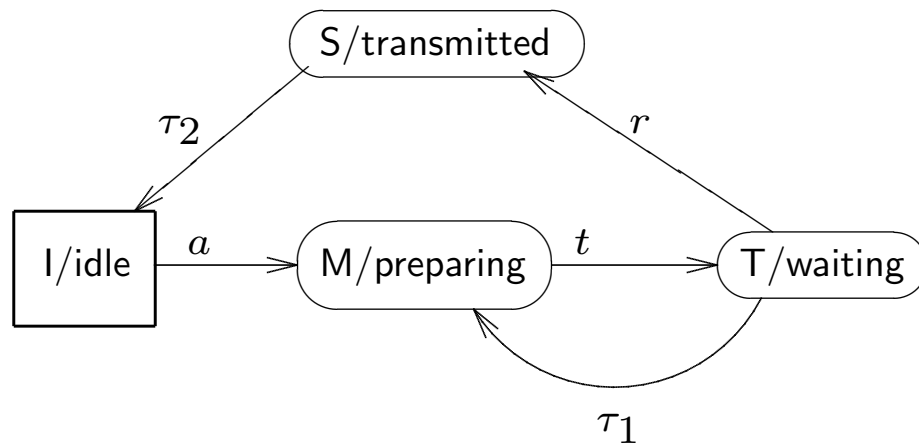
$$(\Sigma, \Delta, X, g, f, x_0)$$

where

- Σ is a finite alphabet
- Δ is a finite **output alphabet**
- X is a finite set of states
- x_0 is the **initial state**, $x_0 \in X$
- g is a **state transition function**, $g : X \times \Sigma \rightarrow X$
- f is an **output function**, $f : X \rightarrow \Delta$

A Moore Machine Example

$M = (\Sigma, \Delta, X, x_0, g, f)$ with



$$\Sigma = \{a, t, \tau_1, \tau_2, r\}$$

$$X = \{I, M, T, S\}$$

$$x_0 = I$$

$$\Delta = \{\text{idle, preparing, waiting, transmitting}\}$$

$$g(I, a) = M, \quad g(M, t) = T$$

$$g(T, r) = S, \quad g(T, \tau_1) = M$$

$$g(S, \tau_2) = I$$

$$f(I) = \text{idle,}$$

$$f(M) = \text{preparing}$$

$$f(T) = \text{waiting,}$$

$$f(S) = \text{transmitted}$$

Mealy Finite State Machines

Definition: A deterministic **Mealy machine** is a six-tuple

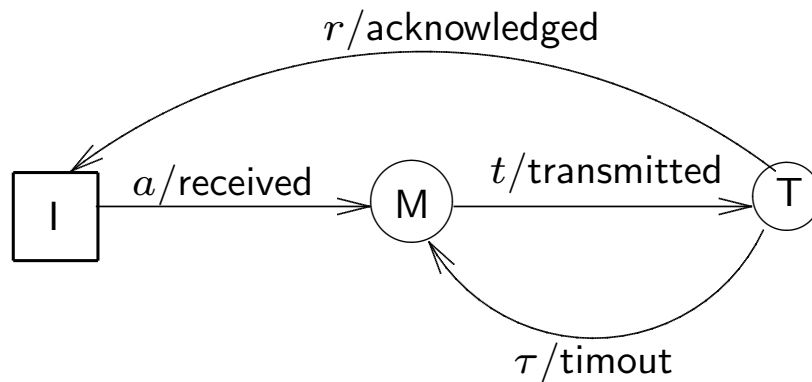
$$(\Sigma, \Delta, X, g, f, x_0)$$

where

- Σ is a finite alphabet
- Δ is a finite **output alphabet**
- X is a finite set of states
- x_0 is the **initial state**, $x_0 \in X$
- g is a **state transition function**, $g : X \times \Sigma \rightarrow X$
- f is an **output function**, $f : X \times \Sigma \rightarrow \Delta$

A Mealy Machine Example

$$M = (\Sigma, \Delta, X, x_0, g, f):$$



$$\Sigma = \{a, t, \tau, r\}$$

$$X = \{I, M, T\}$$

$$x_0 = I$$

$$\Delta = \{\text{received, transmitted, time-out, acknowledged}\}$$

$$g(I, a) = M, \quad g(M, t) = T$$

$$g(T, r) = I, \quad g(T, \tau) = M$$

$$f(I, a) = \text{received,}$$

$$f(M, t) = \text{transmitted}$$

$$f(T, \tau) = \text{time-out,}$$

$$f(S, r) = \text{acknowledged}$$

Summary

- Finite state machines
- State machine with infinite, countable state space
- Nondeterministic FSM
- FSM with ϵ -move
- Equivalence of FSM with nondeterministic FSM with ϵ -FSM
- State equivalence and state aggregation
- Regular expressions and regular languages
- Equivalence of regular languages and FSMs
- Moore FSMs
- Mealy FSMs