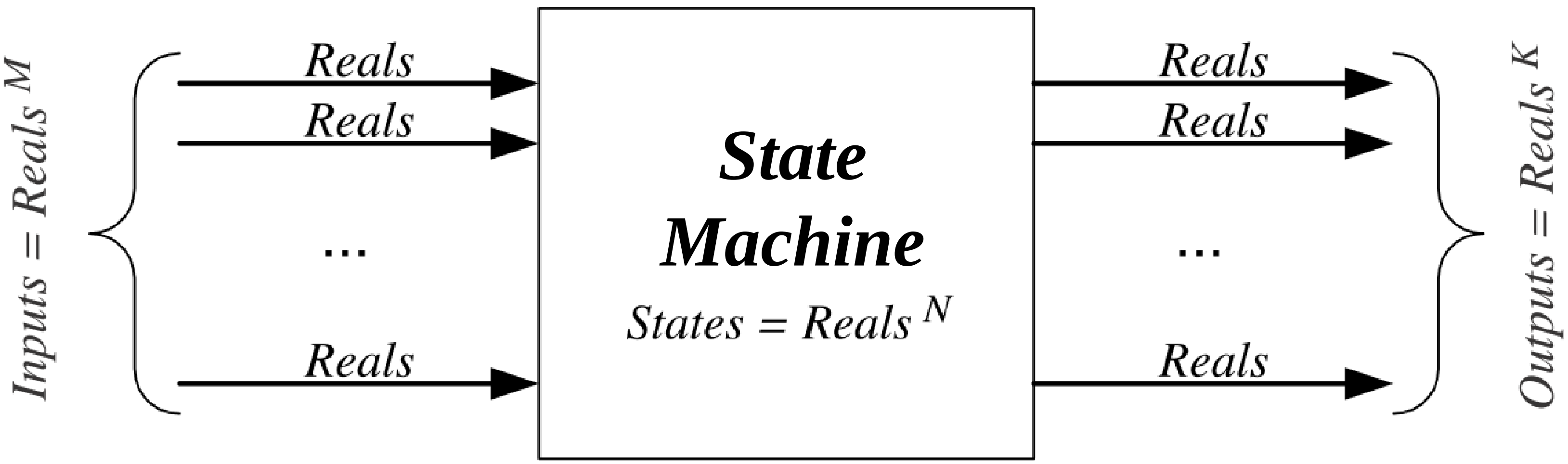
Embedded and Cyber-Physical Systems

- Modeling discrete behavior -

Reference book: Christos G. Cassandras and Stéphane Lafortune, *Introduction to Discrete Event Systems*, Second Edition, Springer, ISBN: 978-0-387-33332-8, 2008.

Discrete model of systems



$$StateMachine = (States, Inputs, Outputs, update, initialState)$$

$$s(n) \quad x(n) \quad y(n) \qquad\qquad s(0)$$

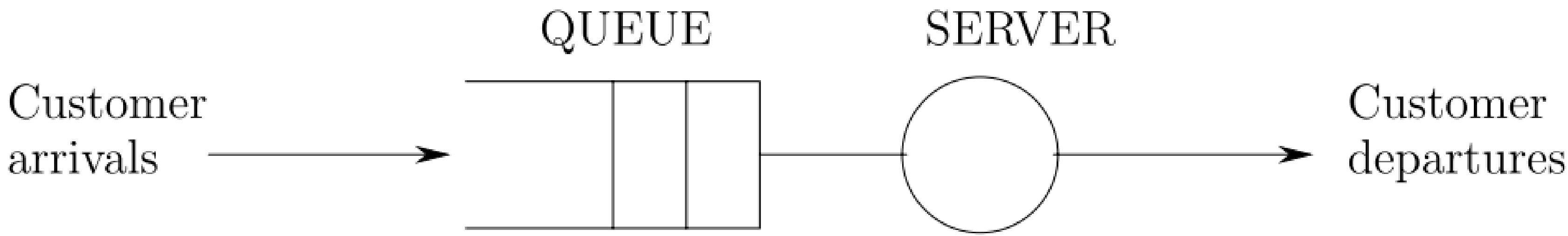
$$\begin{aligned} s(0) &= initialState, \\ \forall n \geq 0,\ (s(n+1), y(n)) &= update(s(n), x(n)) \end{aligned}$$

Discrete Event Systems

- *States* is a discrete (finite or countable) set
- *Inputs* and *outputs* are event sequences
- The *update* function is a transition between discrete states

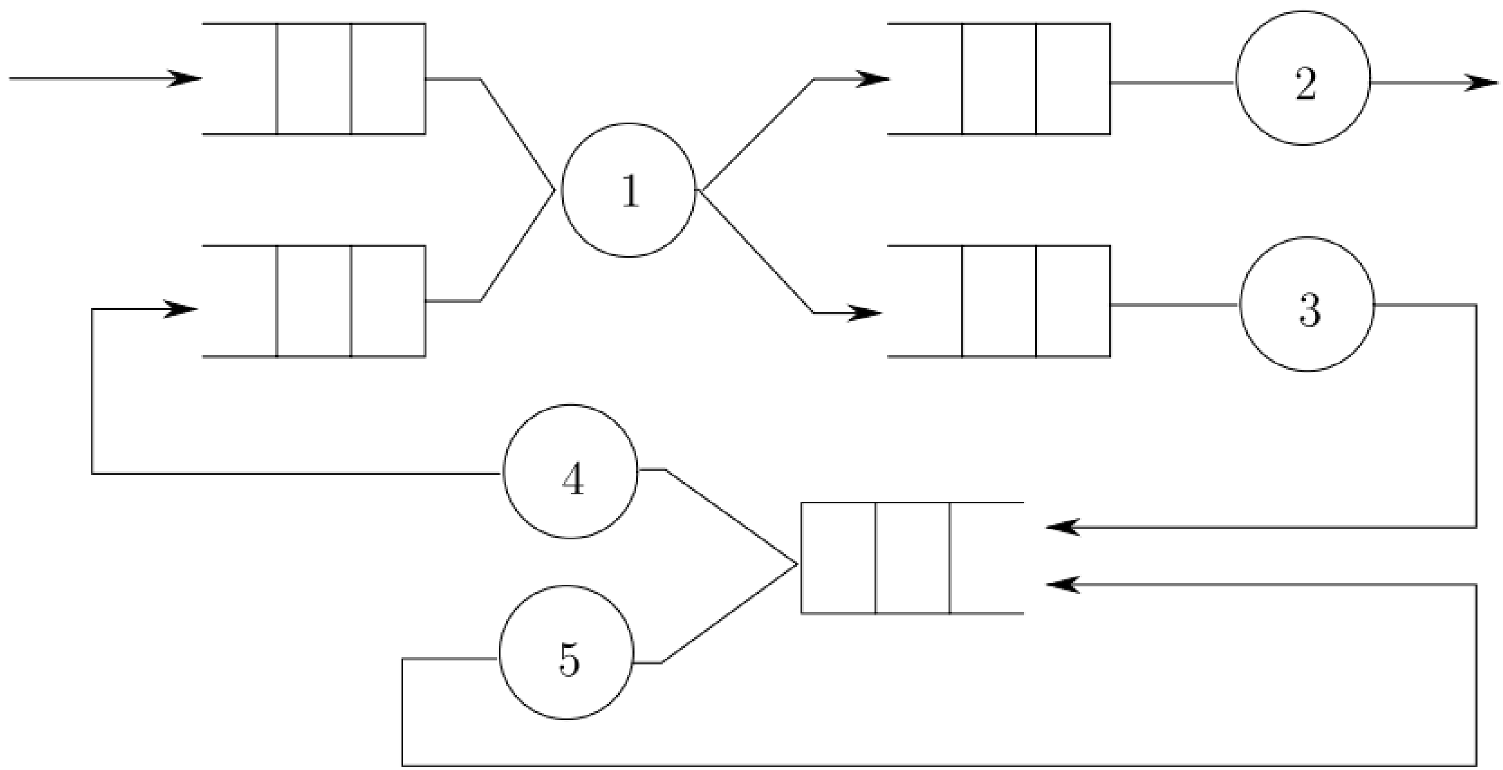
- The system reacts to discrete events, not to the passage of time

Example: Queueing systems



- Elements:
  - Capacity
  - Queueing policy
  - Arrival event
  - Departure event
- What are the states?

Example: Queueing networks

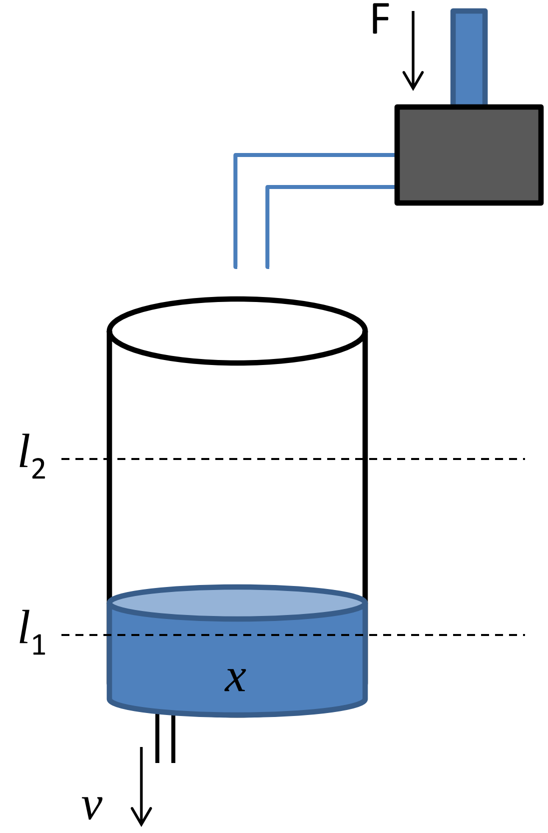


Other examples

- Computer systems (jobs competing for processors and peripherals)
- Manufacturing systems (production parts competing for machines)
- Traffic systems (vehicles competing for space)
- Database systems (maintaining consistency under concurrent transactions)
- *All the above are human artifacts!*

DES as abstractions of continuous systems

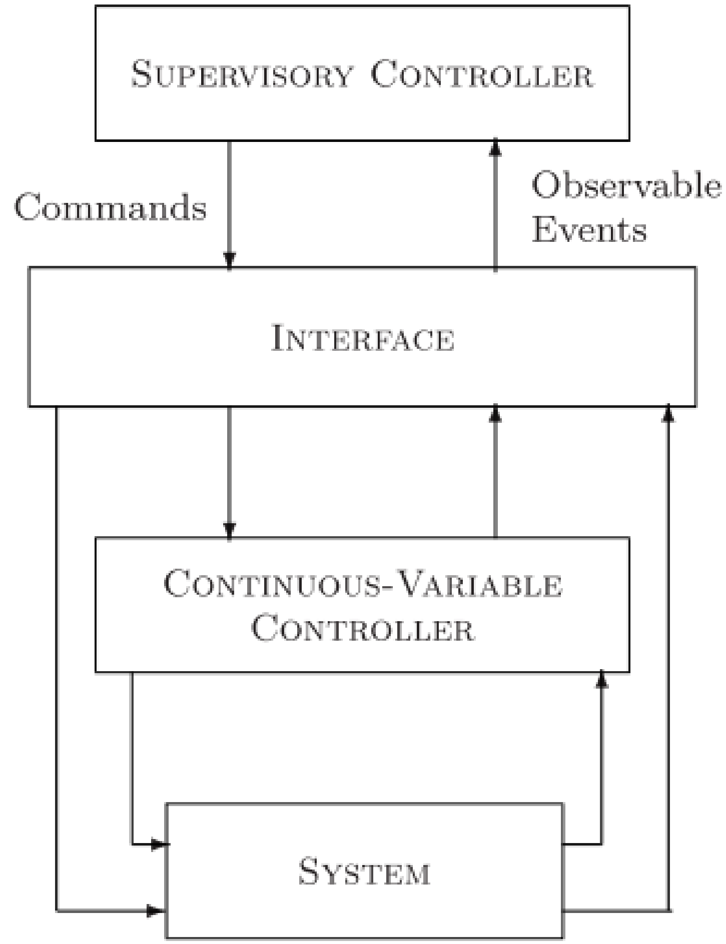
- Another tank example
  - A tank with variable outflow
  - A 2-speed pump
  - Level between l_1 and l_2
  - Minimize load on external supply



- What are the states?
- What are the events?
- Do we need to represent time?

General control architecture

- Continuous control
  - Discrete time
  - Low level
- Discrete control
  - Discrete event
  - Supervisory



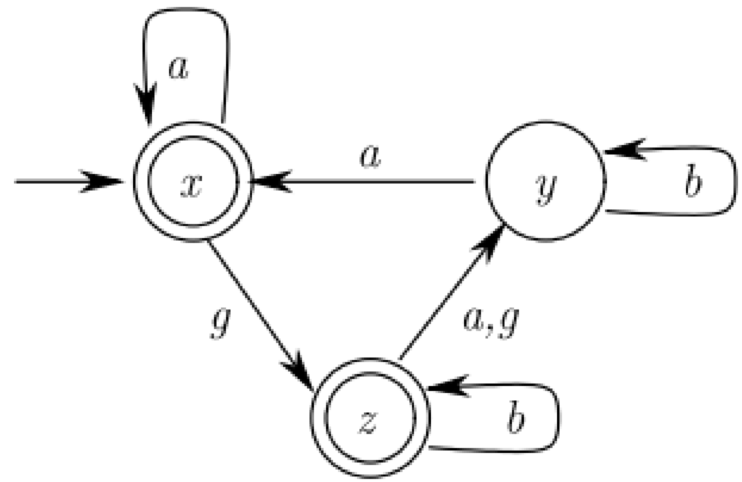
Language models of DES

- Event -> symbol
- Event set -> alphabet
- Event sequence -> word (string)
- System behavior -> language
- Empty string: ε

- What is the language of the tank model?

Representation of languages: Automata

- Event set: E
- State set: X
- Initial state: x_0
- Marked states: X_m
- Transition function $f : X \times E \to X$:



$$f(x,a) = x \qquad f(x,g) = z$$
$$f(y,a) = x \qquad f(y,b) = y$$
$$f(z,b) = z \qquad f(z,a) = f(z,g) = y$$

Deterministic Automaton

$$G = (X, E, f, \Gamma, x_0, X_m)$$

- X is the set of states
- E is a finite set of events
- $f : X \times E \to X$ is the transition function
- $\Gamma : X \to 2^E$ is the active event function
  - $\Gamma(x)$ is the set of all events e for which $f(x,e)$ is defined
- x_0 is the *initial* state
- $X_m \subseteq X$ is the set of *marked states*

Languages represented by Automata

- $E*$ denotes the set of all finite strings of elements in E
- The transition function is extended to strings:

$$\begin{aligned} f(x, \varepsilon) &:= x \\ f(x, se) &:= f(f(x, s), e) \ \text{ for } s \in E^* \text{ and } e \in E \end{aligned}$$

The *language generated* by $G = (X, E, f, \Gamma, x_0, X_m)$ is

$$\mathcal{L}(G) := \{s \in E^* : f(x_0, s) \text{ is defined}\}$$

The *language marked* by G is

$$\mathcal{L}_m(G) := \{s \in \mathcal{L}(G) : f(x_0, s) \in X_m\}$$

Language operations

- Concatenation

$$L_aL_b := \{s \in E^* : (s = s_as_b) \text{ and } (s_a \in L_a) \text{ and } (s_b \in L_b)\}$$

- Prefix-closure

$$\overline{L} := \{s \in E^* : (\exists t \in E^*) \, [st \in L]\}$$

- Kleene-closure

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

- Post-language: Let $L \subseteq E^*$ and $s \in L$. Then

$$L/s := \{t \in E^* : st \in L\}$$

Some exercises

- Show that:

$$\overline{\mathcal{L}(G)} = \mathcal{L}(G)$$

$$\mathcal{L}_m(G) \subseteq \overline{\mathcal{L}_m(G)}$$

$$\overline{\mathcal{L}_m(G)} \subseteq \mathcal{L}(G)$$

Blocking automaton

- An automaton G is *blocking* if

$$\overline{\mathcal{L}_m(G)} \subset \mathcal{L}(G)$$

- Blocking can mean *deadlock* or *livelock*

- G is *nonblocking* when

$$\overline{\mathcal{L}_m(G)} = \mathcal{L}(G)$$

Regular Languages

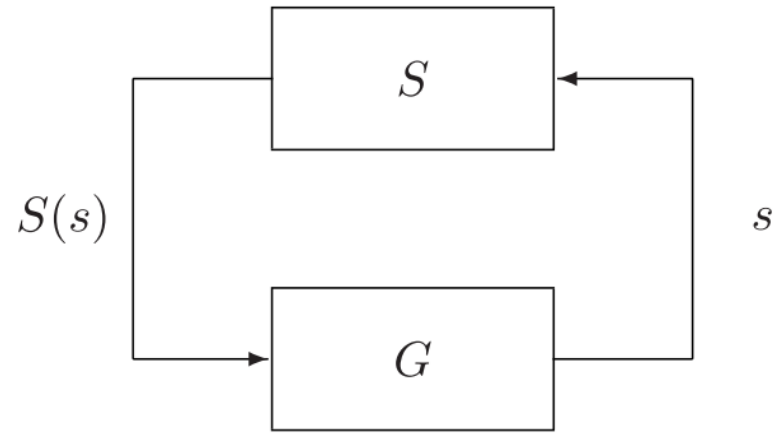
- A language is *regular* if it can be generated by a finite-state automaton.
- The set of regular languages is denoted by $\mathcal{R}$
- Properties:

Let L_1 and L_2 be in $\mathcal{R}$. Then the following languages are also in $\mathcal{R}$:

1. $\overline{L_1}$
2. L_1^*
3. $L_1^c := E^* \setminus L_1$
4. $L_1 \cup L_2$
5. $L_1 L_2$
6. $L_1 \cap L_2$.

Supervisory Control

- G is the uncontrolled DES
- S is a supervisor



- The closed-loop system must satisfy certain specifications such as:
  - Avoid undesirable states in G (e.g., blocking)
  - Some strings in $\mathcal{L}(G)$ are not allowed (illegal sequences)
- A specification is expressed in terms of a legal sublanguage of $\mathcal{L}(G)$

Controller-Plant Interaction

- S observes some events executed by G

$$G = (X, E, f, \Gamma, x_0, X_m) \quad \mathcal{L}(G) = L \quad \text{and} \quad \mathcal{L}_m(G) = L_m$$

- After every observed event, S may *disable* some events in the current active set of G

$$S : \mathcal{L}(G) \rightarrow 2^E$$

- S may not be able to observe all events executed by G (partial observability)

- S may not be able to control all the events of G (partial controllability)

$$E = E_c \cup E_{uc}$$

Closed-loop behavior

- The set of enabled events that G can execute after observing string s is

$$S(s) \cap \Gamma(f(x_0, s))$$

- The language generated by the closed-loop system S/G is defined recursively:

1. $\varepsilon \in \mathcal{L}(S/G)$
2. $[(s \in \mathcal{L}(S/G)) \text{ and } (s\sigma \in \mathcal{L}(G)) \text{ and } (\sigma \in S(s))] \Leftrightarrow [s\sigma \in \mathcal{L}(S/G)]$

- The language marked by S/G is:

$$\mathcal{L}_m(S/G) := \mathcal{L}(S/G) \cap \mathcal{L}_m(G)$$

Specifications of controlled systems

- Most general: required and admissible sublanguages

$$L_r \subseteq \mathcal{L}(S/G) \subseteq L_a \subset \mathcal{L}(G)$$

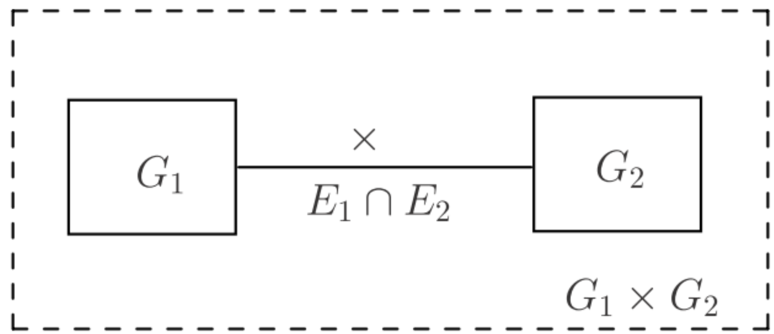
$$L_{rm} \subseteq \mathcal{L}_m(S/G) \subseteq L_{am} \subset \mathcal{L}_m(G)$$

- For regular languages, the specification is translated into an automaton H_{spec}

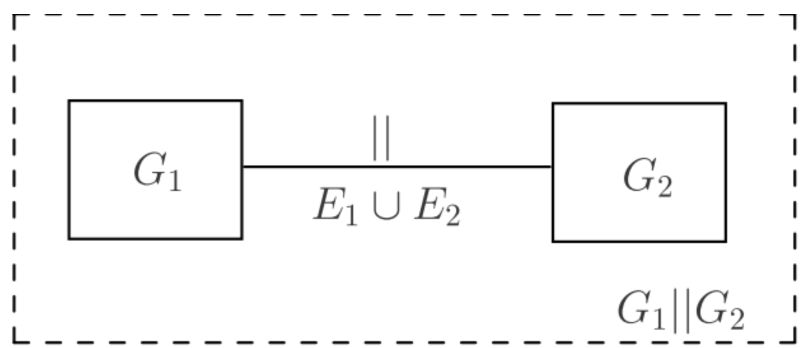
- The admissible language L_a is the language generated by automaton H_a, which is a composition of H_{spec} and G

Composition of Automata

- Formally expresses interconnections between subsystems
- Product:



- Parallel:



Product of Automata

$$G_1 = (X_1, E_1, f_1, \Gamma_1, x_{01}, X_{m1}) \qquad G_2 = (X_2, E_2, f_2, \Gamma_2, x_{02}, X_{m2})$$

- The product of G_1 and G_2 is the automaton

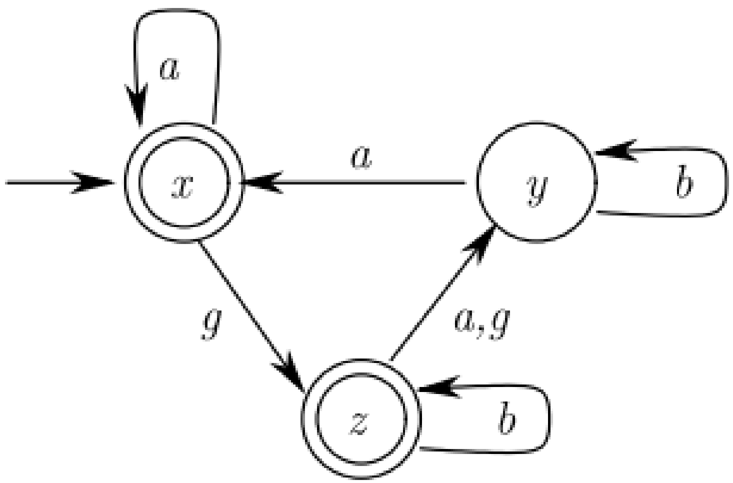
$$G_1 \times G_2 := Ac(X_1 \times X_2, E_1 \cup E_2, f, \Gamma_{1\times 2}, (x_{01}, x_{02}), X_{m1} \times X_{m2})$$

- where

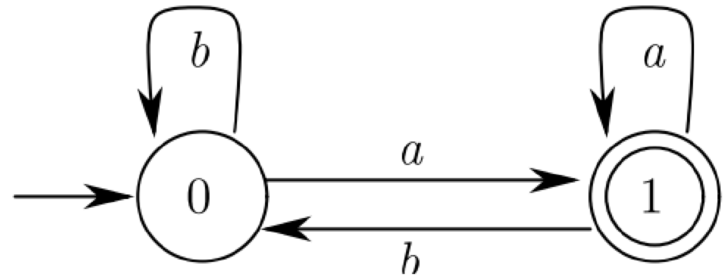
$$f((x_1, x_2), e) := \begin{cases} (f_1(x_1, e), f_2(x_2, e)) & \text{if } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ \text{undefined} & \text{otherwise} \end{cases}$$

- An event occurs in the product if and only if it occurs in both G_1 and G_2 (lock-step operation)
- What is the language generated by G_1 x G_2 ?

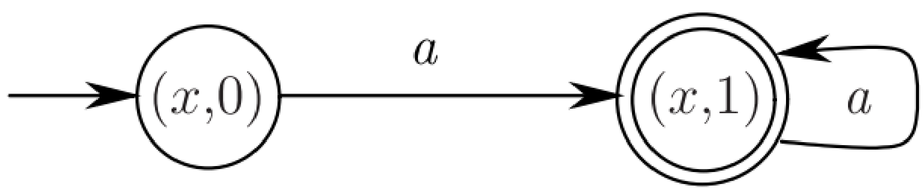
Product example



X



=



Parallel composition of automata

$$G_1 = (X_1, E_1, f_1, \Gamma_1, x_{01}, X_{m1}) \qquad G_2 = (X_2, E_2, f_2, \Gamma_2, x_{02}, X_{m2})$$

- The parallel composition of G_1 and G_2 is the automaton

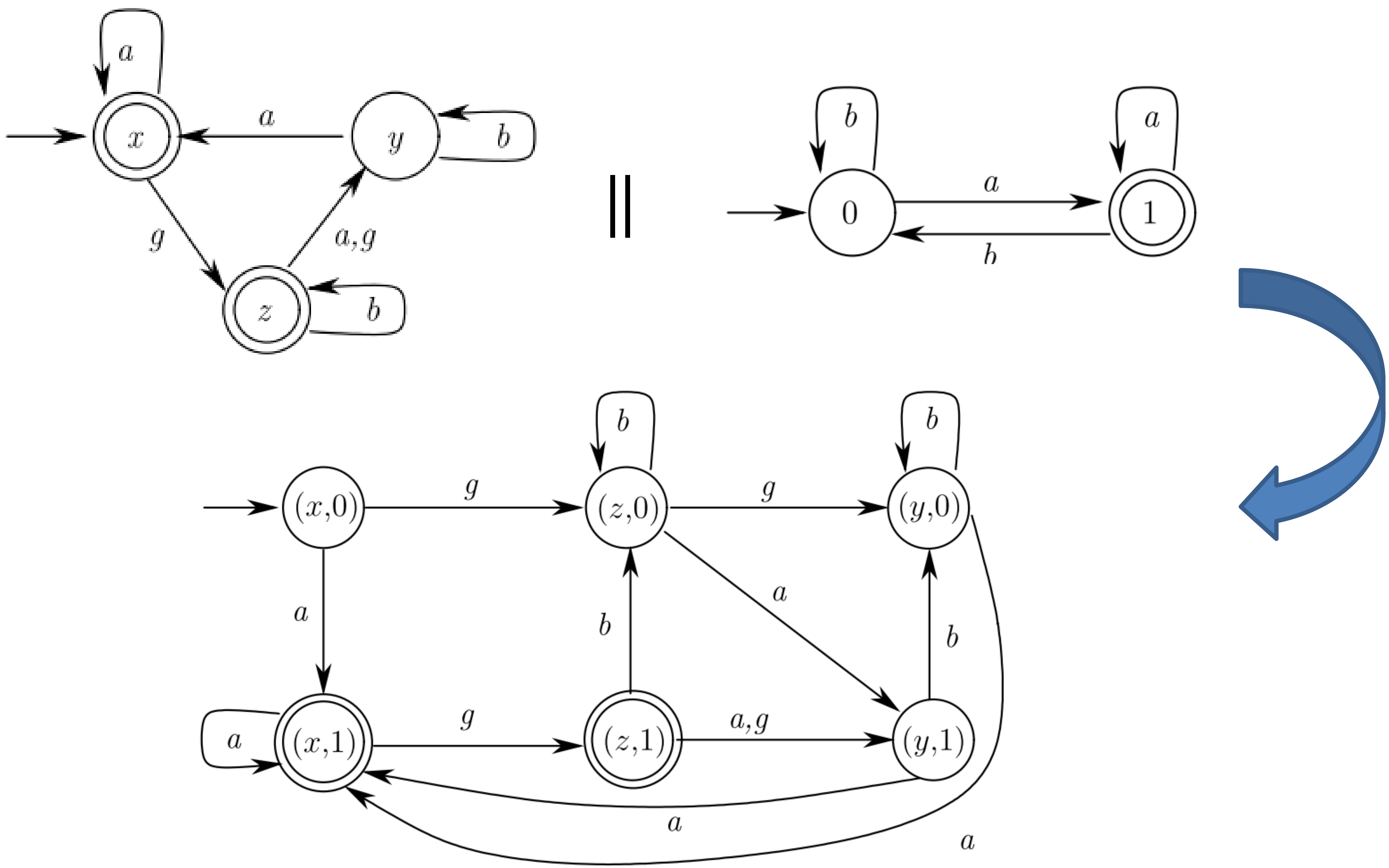
$$G_1 \mid\mid G_2 := Ac(X_1 \times X_2, E_1 \cup E_2, f, \Gamma_{1||2}, (x_{01}, x_{02}), X_{m1} \times X_{m2})$$

where

$$f((x_1, x_2), e) := \begin{cases} (f_1(x_1, e), f_2(x_2, e)) & \text{if } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ (f_1(x_1, e), x_2) & \text{if } e \in \Gamma_1(x_1) \setminus E_2 \\ (x_1, f_2(x_2, e)) & \text{if } e \in \Gamma_2(x_2) \setminus E_1 \\ \text{undefined} & \text{otherwise} \end{cases}$$

- Automata are synchronized on common events only
- Allows each automaton to operate individually on private events

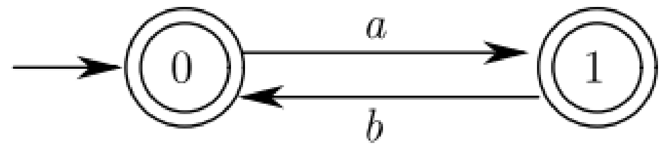
Example of parallel composition



Examples of specifications

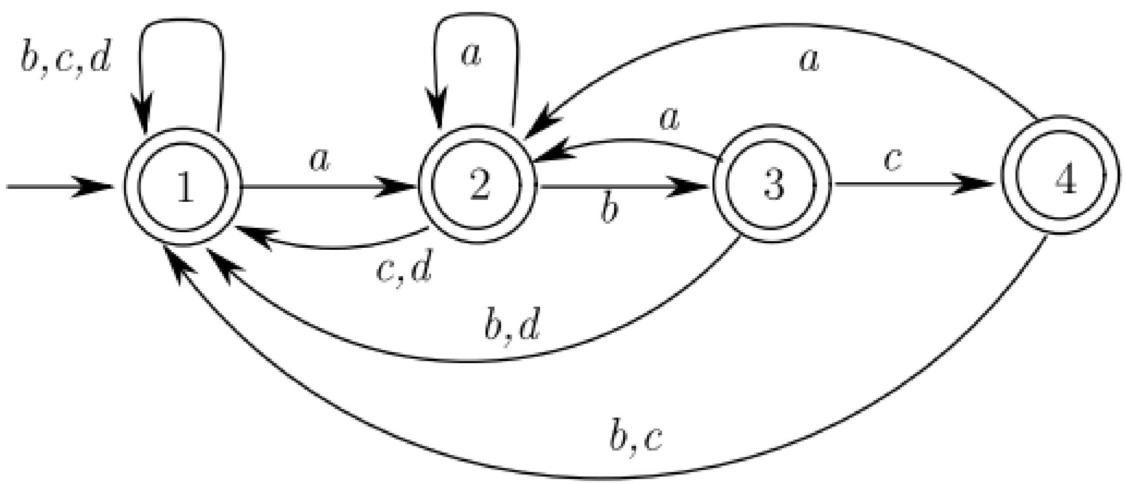
- Event alternation: Events a and b must occur alternatively, with a first

H_{spec} :



- Illegal substring: Event sequence $abcd$ must not occur

H_{spec} :

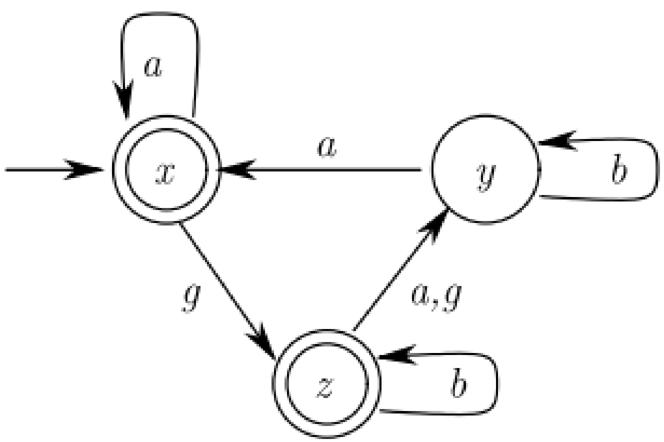


Admissible language

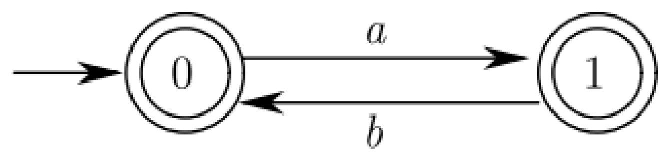
- Generated by the parallel composition of the uncontrolled system G and the specification model:

  $H_a = G \,||\, H_{spec}$

- Example: H_a = ?



G



H_{spec}