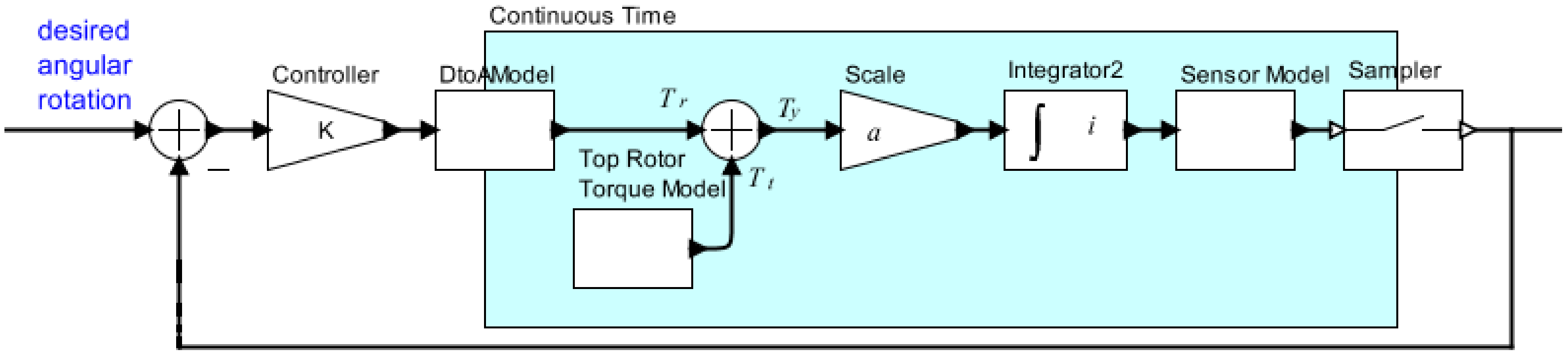
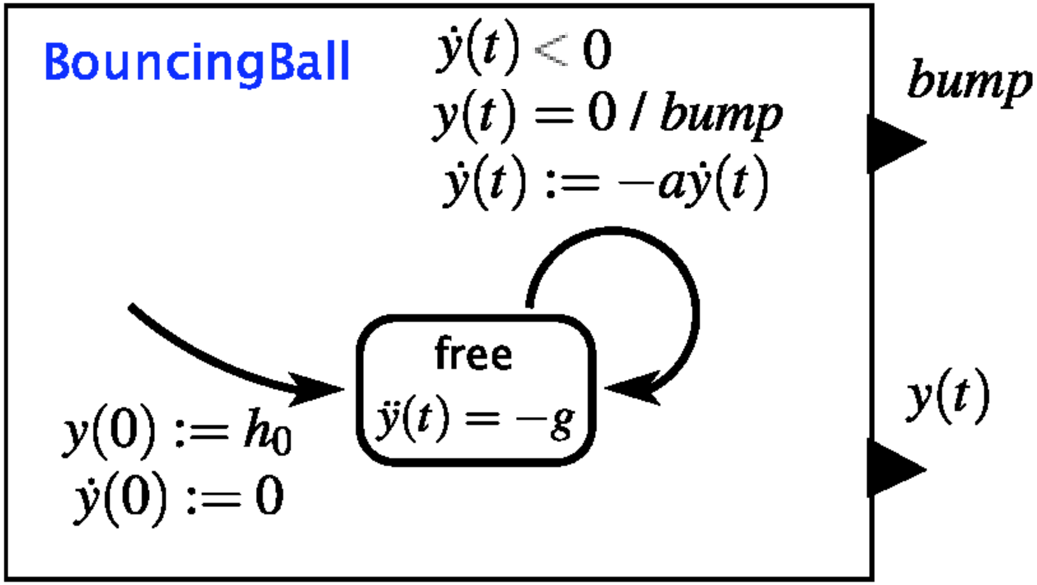
Embedded Systems 5

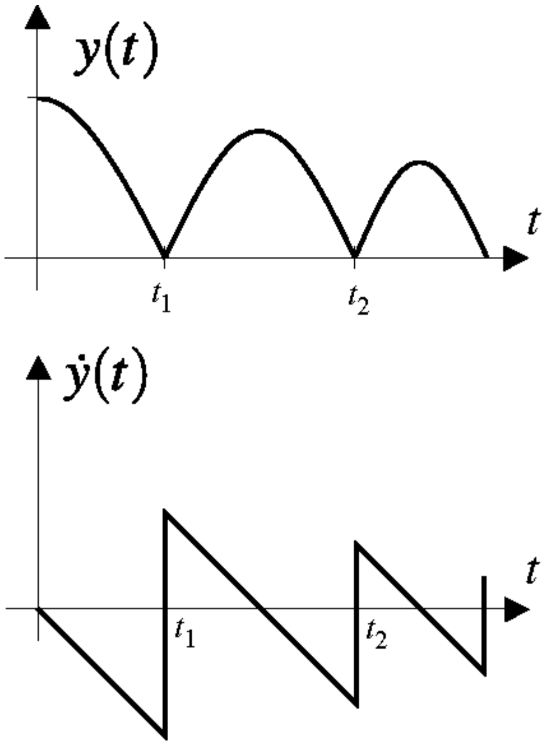
Mixed Signal Models



The signals inside the blue area are continuous-time signals, and the ones outside are discrete-time signals.

REVIEW: Hybrid Automaton for Bouncing Ball





y – vertical distance from ground (position)

a – coefficient of restitution, $0 < a < 1$

Hybrid Automata

- Q: set of modes
- S: set of state variables, partitioned into
  - $C=\{c_1, c_2, ..., c_n\}$: continuous signals (with range $\Re$)
  - $D=\{d_1, d_2, ..., d_m\}$: discrete signals (with range $\{absent\} \cup X$)
- $U=\{u_1, u_2, ..., u_k\}$: set of input signals,
- $Init \subseteq Q \times \Re^n \times (\{absent\} \cup X)^m$: initial condition
- F: flows, defining differential equations for each continuous state variable in each mode
- J: $Q \times Guards \rightarrow Q \times Resets$: jumps, where Guards is a constraint over C and U and Resets is a set of assignments of the form $x_i := expr(X,U)$ for the state variables

Note: our definitions follow Lee/Seshia, there are several other definitions of hybrid automata

Hybrid Time Set

A hybrid time set is a finite or infinite sequence of intervals $\tau = \{I_i\}_{i=0..N}$ such that

- $I_i = [\tau_i, \tau_i']$ for all $i < N$;
- If $N < \infty$ then either $I_N = [\tau_N, \tau_N']$ or $I_N = [\tau_N, \tau_N')$; and
- $\tau_i \leq \tau_i' = \tau_{i+1}$ for all i

Hybrid Trajectory

- A hybrid trajectory (τ, q, x) consists of a hybrid time set τ and two sequences of functions
  - $q = \{q_i(\cdot): I_i \rightarrow Q\}_{i=0..N}$
  - $x = \{c_i(\cdot): I_i \rightarrow \Re\}_{i=0..N} \cup \{d_i(\cdot): I_i \rightarrow X \cup \{absent\}\}_{i=0..N}$

Execution of a Hybrid Automaton

time

$$
\begin{array}{ll}
\tau_0 & (q_0, \mathbf{x}_0) \\
\tau_0' & \rightsquigarrow (q_0, \mathbf{x}_0') \\
= & \downarrow \\
\tau_1 & (q_1, \mathbf{x}_1) \\
\tau_1' & \rightsquigarrow (q_1, \mathbf{x}_1') \\
 & \downarrow \\
 & \vdots \\
 & \downarrow \\
\tau_N & (q_N, \mathbf{x}_N) \\
\tau_N' & \rightsquigarrow (q_N, \mathbf{x}_N') \\
 & \downarrow \\
 & \vdots
\end{array}
$$

An **execution** of a hybrid automaton is a hybrid trajectory (τ, q, x) that statisfies the following conditions

- Initial condition: $(q_0, x_0) \in$ Init
- Discrete evolution: the pair $((q_i(\tau'_i),x_i(\tau'_i)), (q_{i+1}(\tau_{i+1}),x_i(\tau_{i+1}))$ satisfies J
- Continuous evolution: for all i,
    1. $q_i(\cdot)$ is constant over I_i
    2. $c_i(\cdot)$ is the solution to the differential equations in $F(q(\tau_i))$
    3. $d_i(\cdot)$ are absent during (τ_i,τ'_i)
    4. All jumps in J are disabled during (τ_i,τ'_i)

Execution of a Hybrid Automaton

time ↓

τ_0	$(q_0, \mathbf{x}_0)$
τ'_0	$\rightsquigarrow (q_0, \mathbf{x}'_0)$
$=$	$\downarrow$
τ_1	$(q_1, \mathbf{x}_1)$
τ'_1	$\rightsquigarrow (q_1, \mathbf{x}'_1)$
	$\downarrow$
	$\vdots$
	$\downarrow$
τ_N	$(q_N, \mathbf{x}_N)$
τ'_N	$\rightsquigarrow (q_N, \mathbf{x}'_N)$
	$\downarrow$
	$\vdots$

$$\tau = \tau_0, \tau_1, \tau_2, \ldots, \tau_N [, \ldots]$$

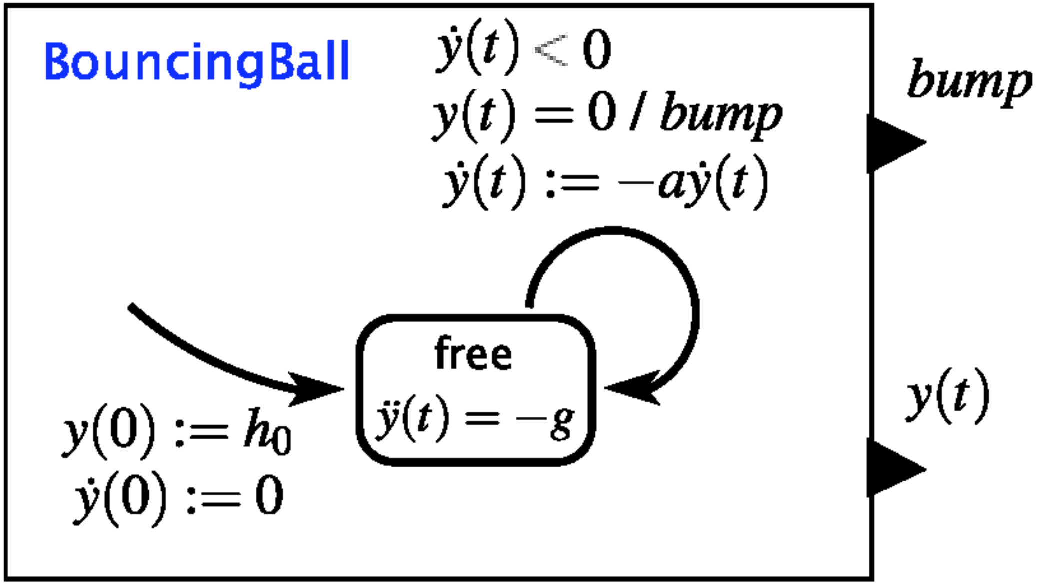
Continuous extent of τ:

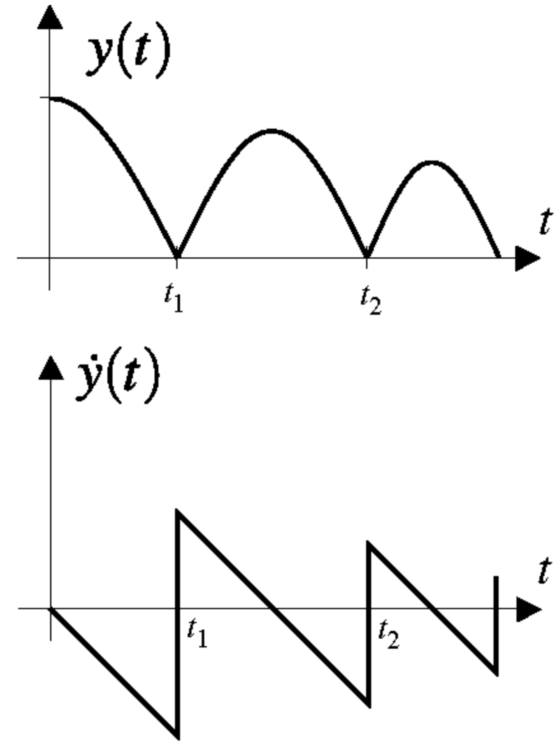
$$|\tau| = \sum_{i=0}^{\infty} \tau_{i+1} - \tau_i$$

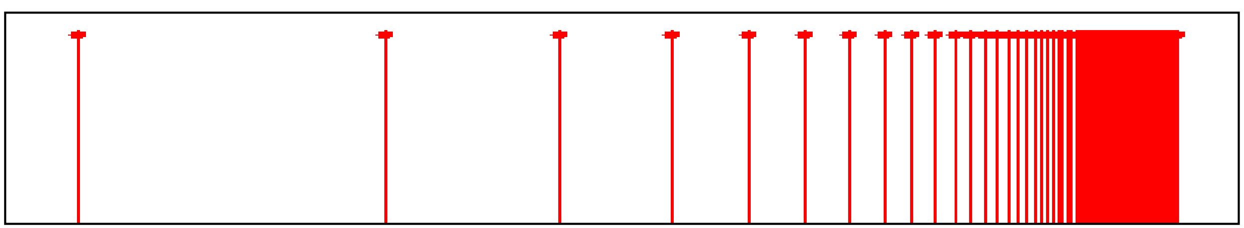
Discrete extent of τ:

$$\langle\tau\rangle = \begin{cases} N & \text{if } \tau \text{ is a finite sequence of length } N \\ \infty & \text{if } \tau \text{ is an infinite sequence} \end{cases}$$

REVIEW: Hybrid Automaton for Bouncing Ball







Zeno Behavior

time

$$
\begin{array}{ll}
\tau_0 & (q_0, \mathbf{x}_0) \\
\tau_0' & \rightsquigarrow (q_0, \mathbf{x}_0') \\
= & \downarrow \\
\tau_1 & (q_1, \mathbf{x}_1) \\
\tau_1' & \rightsquigarrow (q_1, \mathbf{x}_1') \\
 & \downarrow \\
 & \vdots \\
 & \downarrow \\
\tau_N & (q_N, \mathbf{x}_N) \\
\tau_N' & \rightsquigarrow (q_N, \mathbf{x}_N') \\
 & \downarrow \\
 & \vdots
\end{array}
$$

An execution of a hybrid automaton with time set τ is **zeno** iff $\langle \tau \rangle = \infty$ but $|\tau| < \infty$.

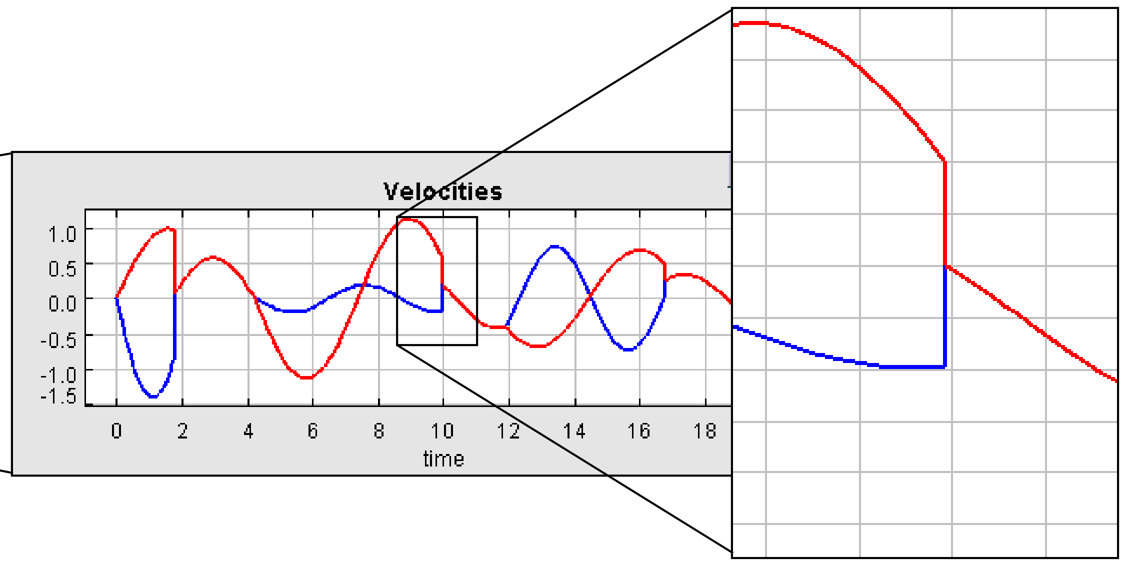
Superdense Time

A signal can have a sequence of values at each (real) time.

$$\mathbf{x} : \mathbb{R} \times \mathbb{N} \to \mathbb{R}^3$$

$$\dot{\mathbf{x}} : \mathbb{R} \times \mathbb{N} \to \mathbb{R}^3$$

$$\ddot{\mathbf{x}} : \mathbb{R} \times \mathbb{N} \to \mathbb{R}^3$$



At (real) time t, x has a sequence of values

$$x(t, 0), x(t, 1), \cdots$$

Initial and final value signals

Let $x\colon \mathbb{R} \times \mathbb{Z} \to \mathbb{R}^3$ be a CT signal. Define the *initial value signal* to be a function $x_i\colon \mathbb{R} \to \mathbb{R}$ where

$$x_i(t) = x(t, 0)$$

Define the *final value signal* to be a function $x_f\colon \mathbb{R} \to \mathbb{R}$ where

$$x_f(t) = x(t, m)$$

where $m \in \mathbb{N}$ is the least value such that

$$\forall\, n > m, \quad x(t, n) = x(t, m).$$

If there is no such m at any t, then the signal is said to be a *stuttering Zeno signal*.

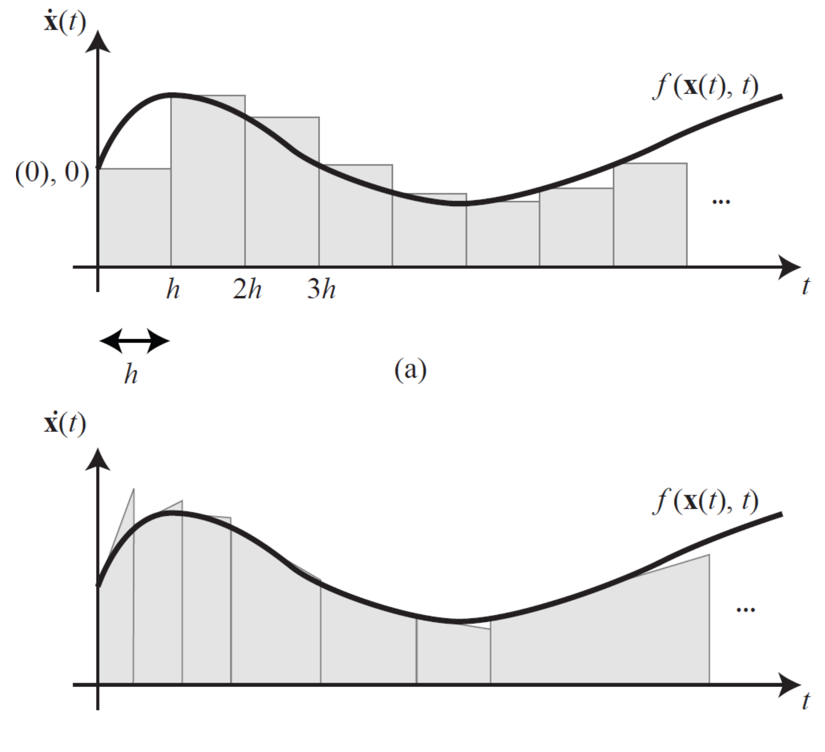
Simulation of continuous-time systems

- The (numeric) simulator cannot directly deal with the time continuum, but can approximate it
- We consider equations of the form

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), t)$$

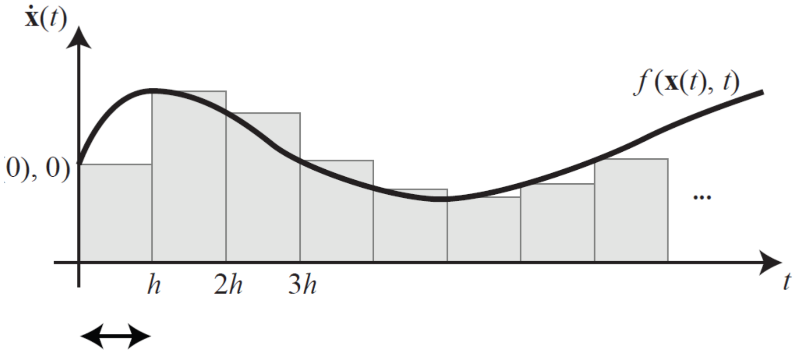
- An equivalent model is an integral equation

$$\begin{aligned} \mathbf{x}(t) &= \mathbf{x}(0) + \int_0^t \dot{\mathbf{x}}(\tau) d\tau \\ &= \mathbf{x}(0) + \int_0^t f(\mathbf{x}(\tau), \tau) d\tau \end{aligned}$$



(a)
(b)

Forward Euler Solver



(a)

A forward Euler solver estimates the value of **x** at time points $0, h, 2h, 3h, \cdots$, where h is called the **step size**. The integration is approximated as follows,

$$\begin{aligned}
\mathbf{x}(h) &= \mathbf{x}(0) + hf(\mathbf{x}(0), 0) \\
\mathbf{x}(2h) &= \mathbf{x}(h) + hf(\mathbf{x}(h), h) \\
\mathbf{x}(3h) &= \mathbf{x}(2h) + hf(\mathbf{x}(2h), 2h) \\
&\cdots \\
\mathbf{x}((k+1)h) &= \mathbf{x}(kh) + hf(\mathbf{x}(kh), kh).
\end{aligned}$$

PETRI NETS

Petri nets

Introduced in 1962 by Carl Adam Petri

Application areas:

- modelling, analysis, verification of distributed systems
- automation engineering
- business processes
- modeling of resources
- modeling of synchronization

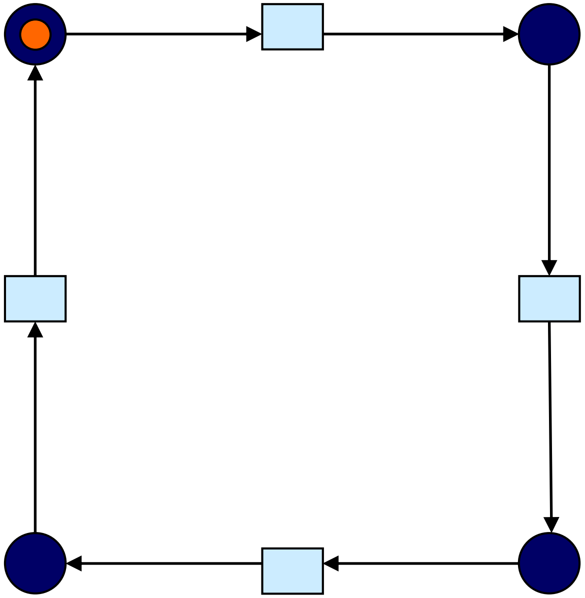
Focus on modeling causal dependencies;

no global synchronization assumed (message passing only).

Concurrency and parallelism

- Concurrency is central to embedded systems. A computer program is said to be **concurrent** if different parts of the program conceptually execute simultaneously.
- A program is said to be **parallel** if different parts of the program physically execute simultaneously on distinct hardware (multi-core, multi-processor or distributed systems)

Example 1: The four seasons

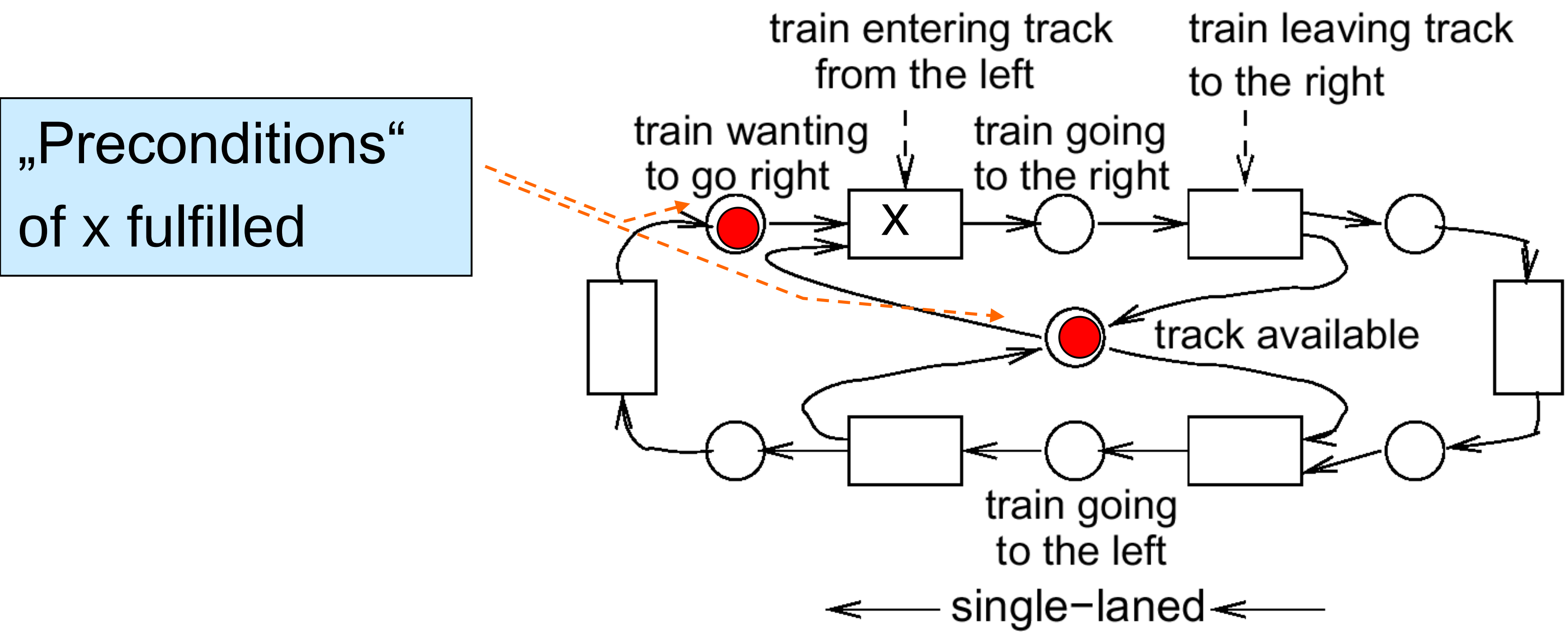


Key Elements

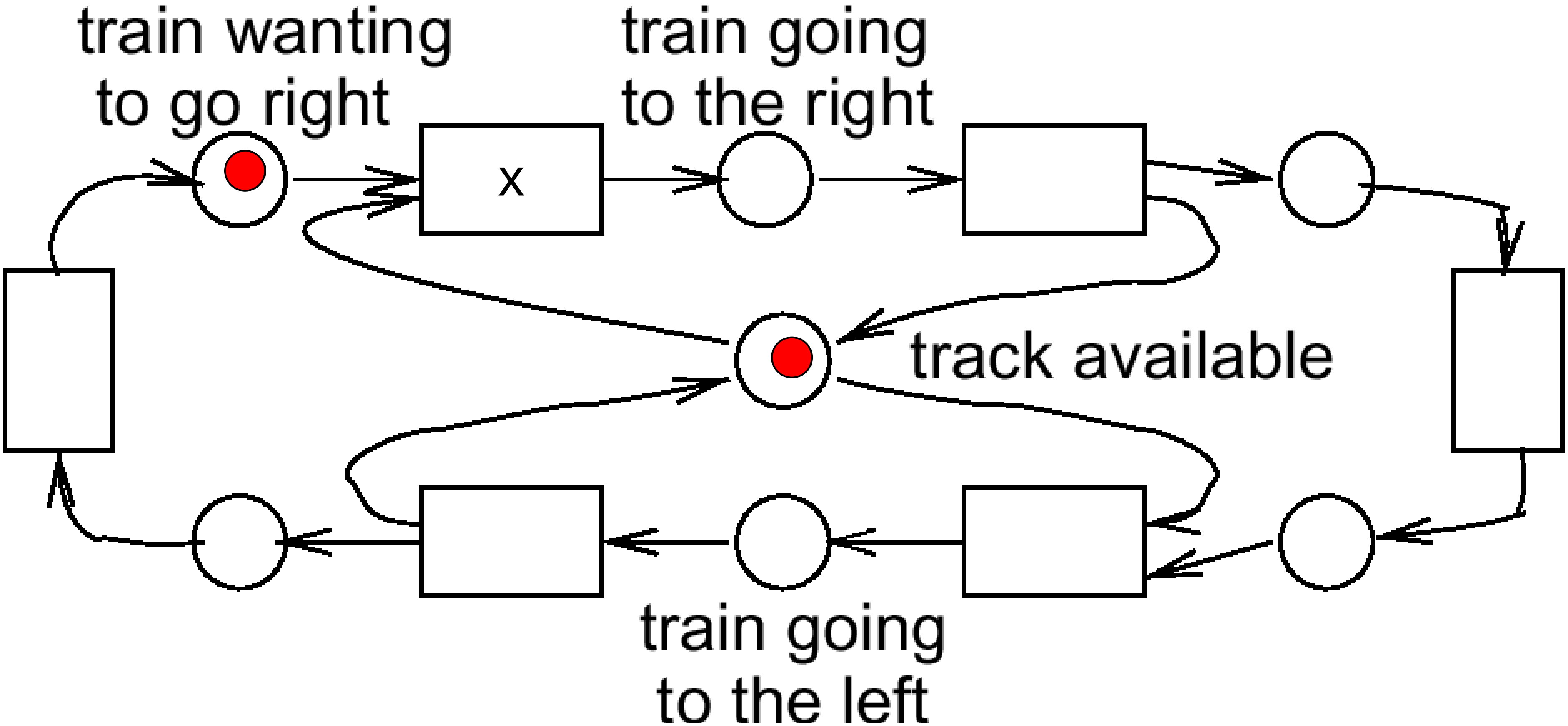
- **Conditions**
  Either met or not met. Conditions represent “local states”. Set of conditions describes the potential state space.
- **Events**
  May take place if certain conditions are met. Event represents a state transition.
- **Flow relation**
  Relates conditions and events, describes how an event changes the local and global state.
- **Tokens**
  Assignments of tokens to conditions specifies a global state.

Example 2:
Synchronization at single track rail segment

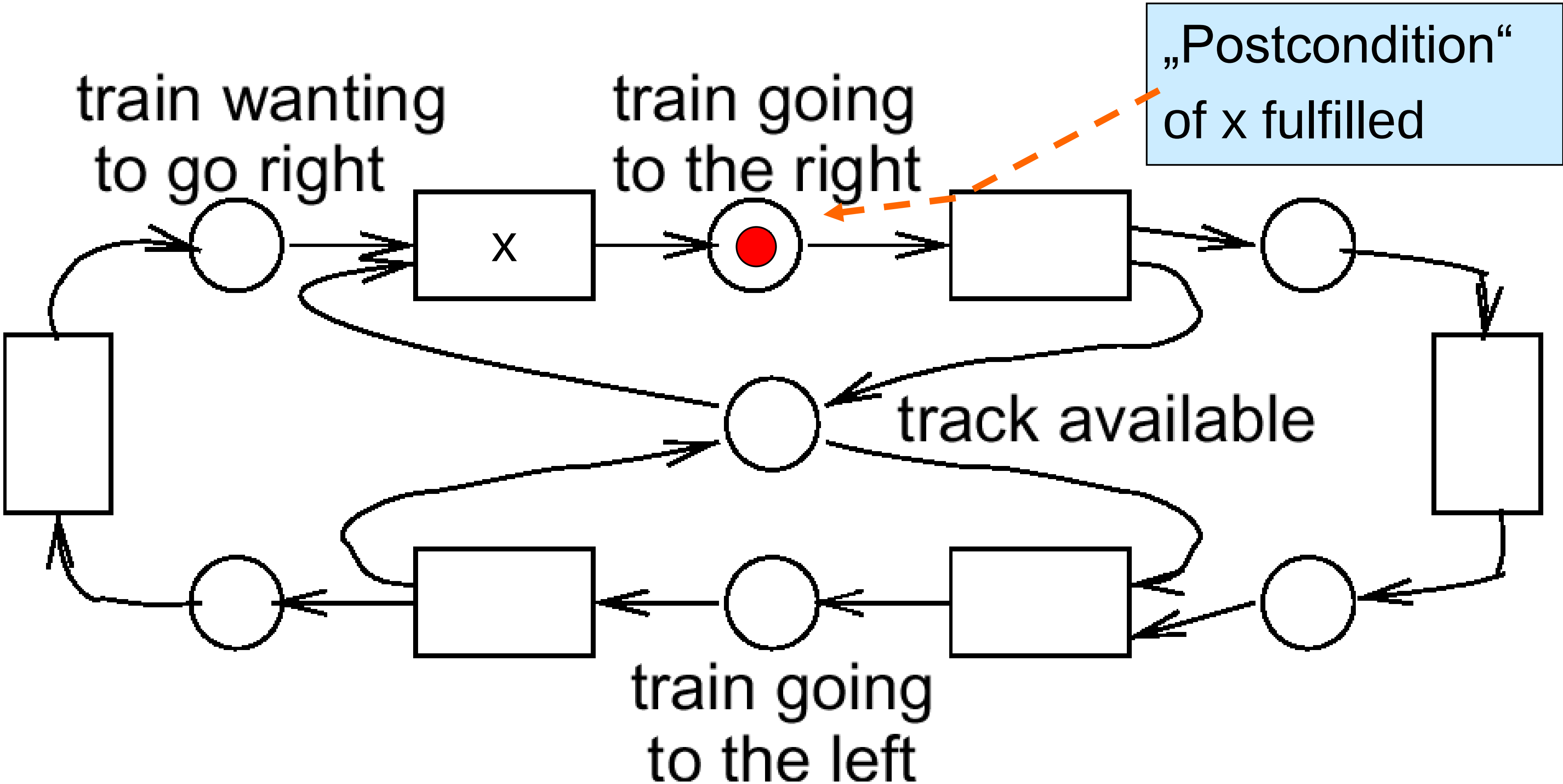
- **mutual exclusion:**
  there is at most one train using the track rail



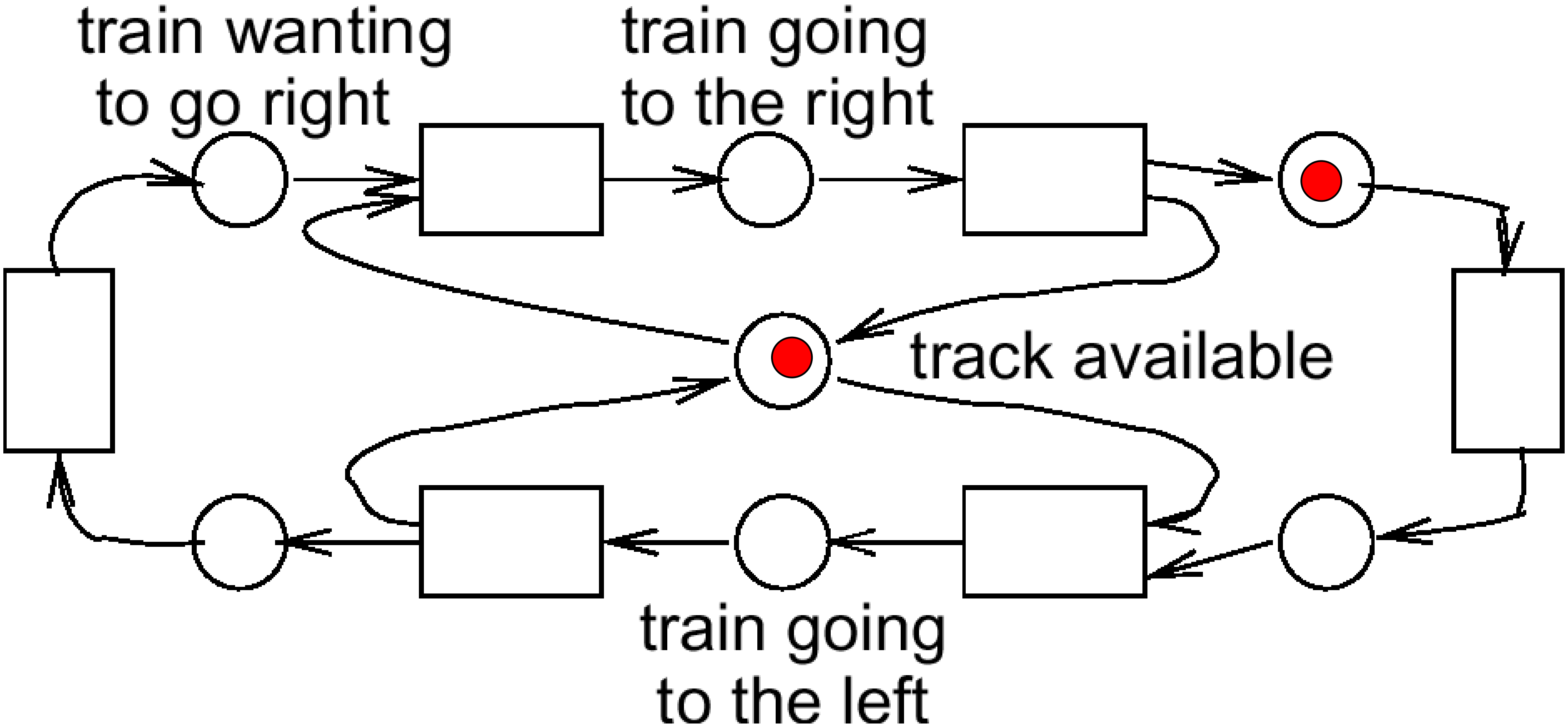
Playing the „token game“: dynamic behavior



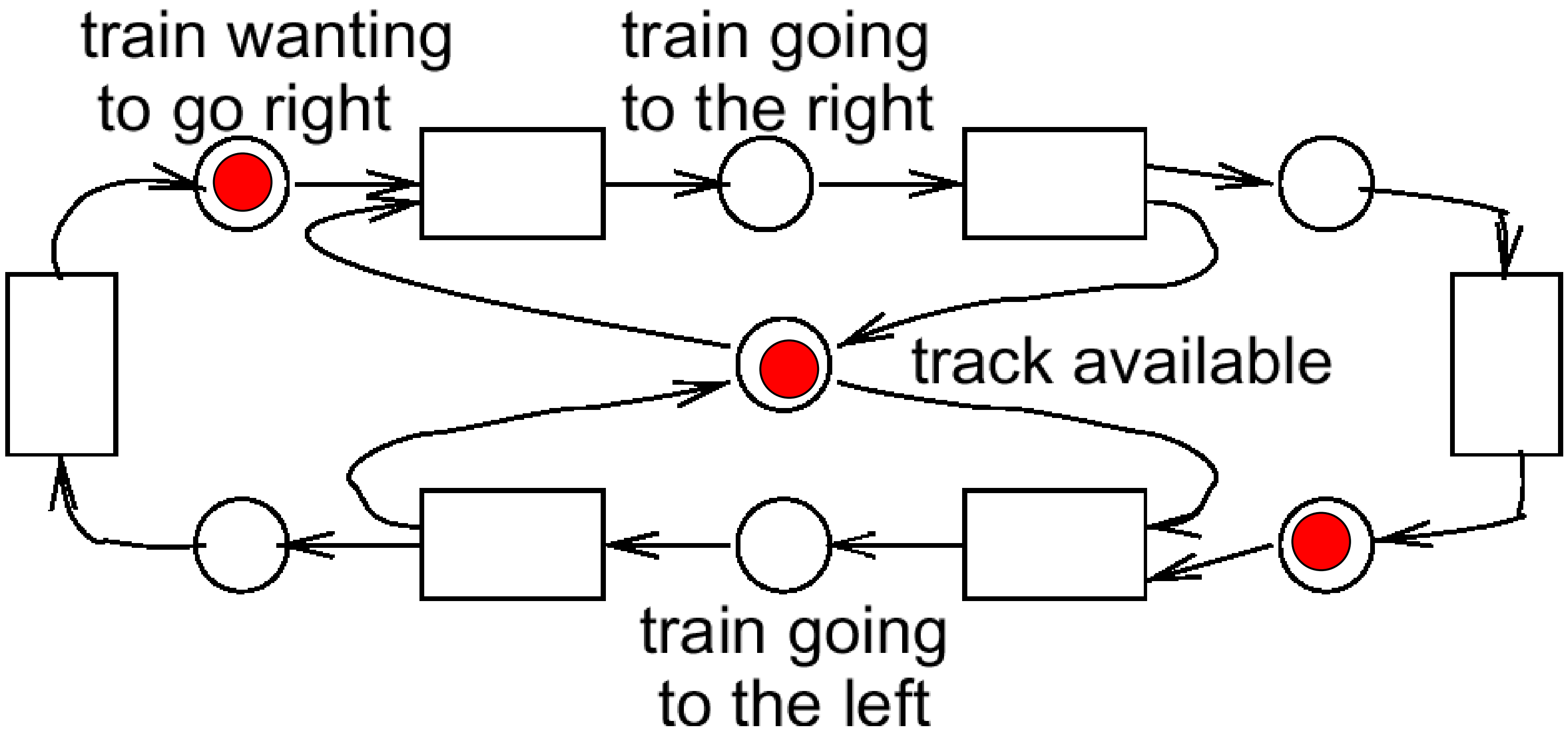
Playing the „token game“: dynamic behavior



Playing the „token game“: dynamic behavior



Conflict for resource „track“: two trains competing



Condition/event Petri nets

single token per place

Def.: $N=(C,E,F)$ is called a **Petri net**, iff the following holds

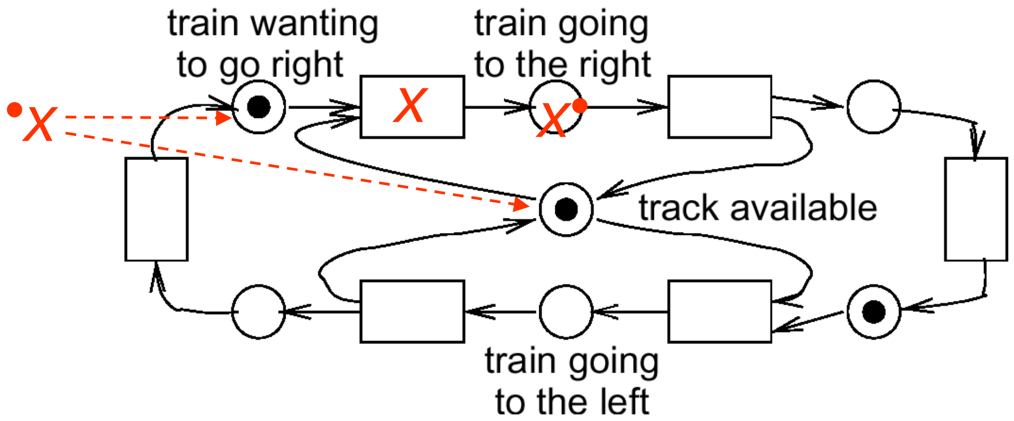
1. C and E are disjoint sets
2. $F \subseteq (C \times E) \cup (E \times C)$; is binary relation, („**flow relation**“)

Def.: Let N be a net and let $x \in (C \cup E)$.

$^{\bullet}x := \{y \mid y\ F\ x\}$ is called the set of **preconditions.**

$x^{\bullet} := \{y \mid x\ F\ y\}$ is called the set of **postconditions.**

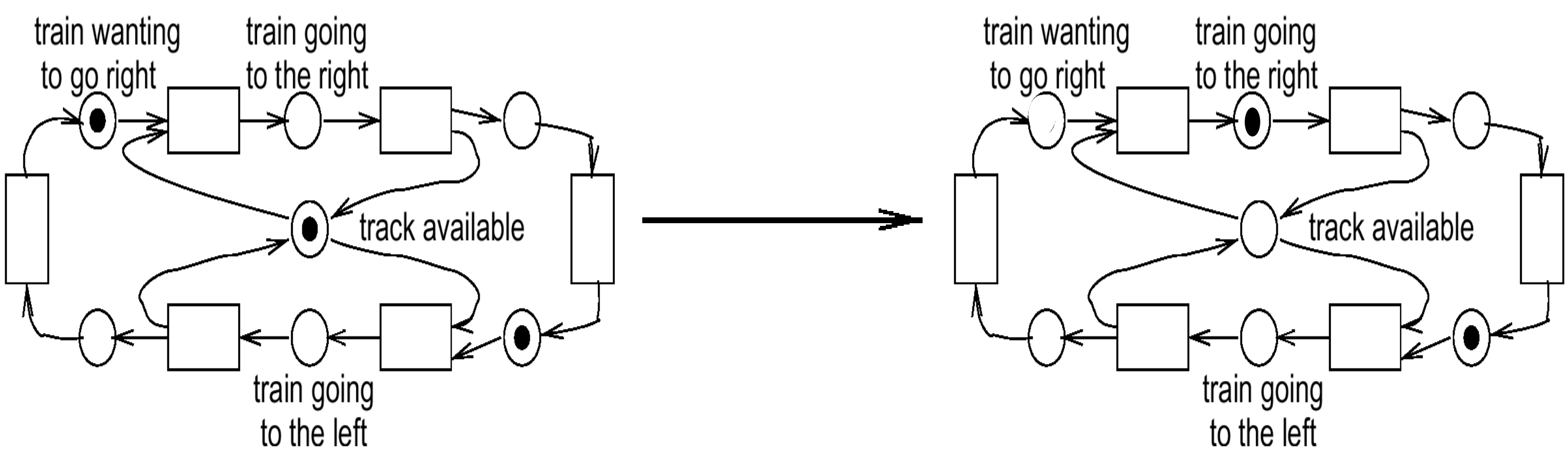
Example:



Boolean marking and computing changes of markings

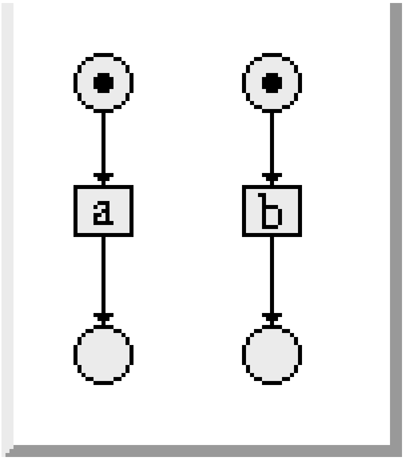
- A Boolean marking is a mapping M: C → { 0,1 }.
- „Firing“ events *x* generate new markings on each of the conditions *c* according to the following rules:

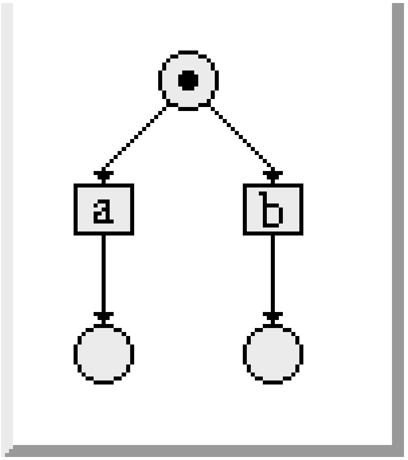
  a transition at *x* can be fired, iff •*x*, *i.e.* all preconditions of *x* are marked and *x*• is not marked, after firing •*x is* unmarked and *x*• is marked
- M → M’, iff M’ results from M by firing exactly one transition

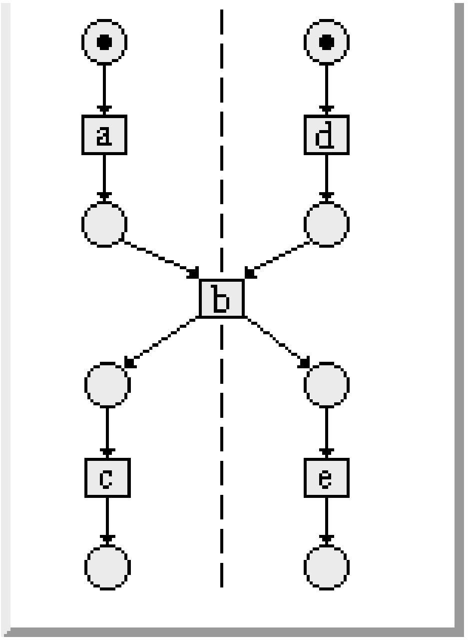


Expressiveness: basic examples

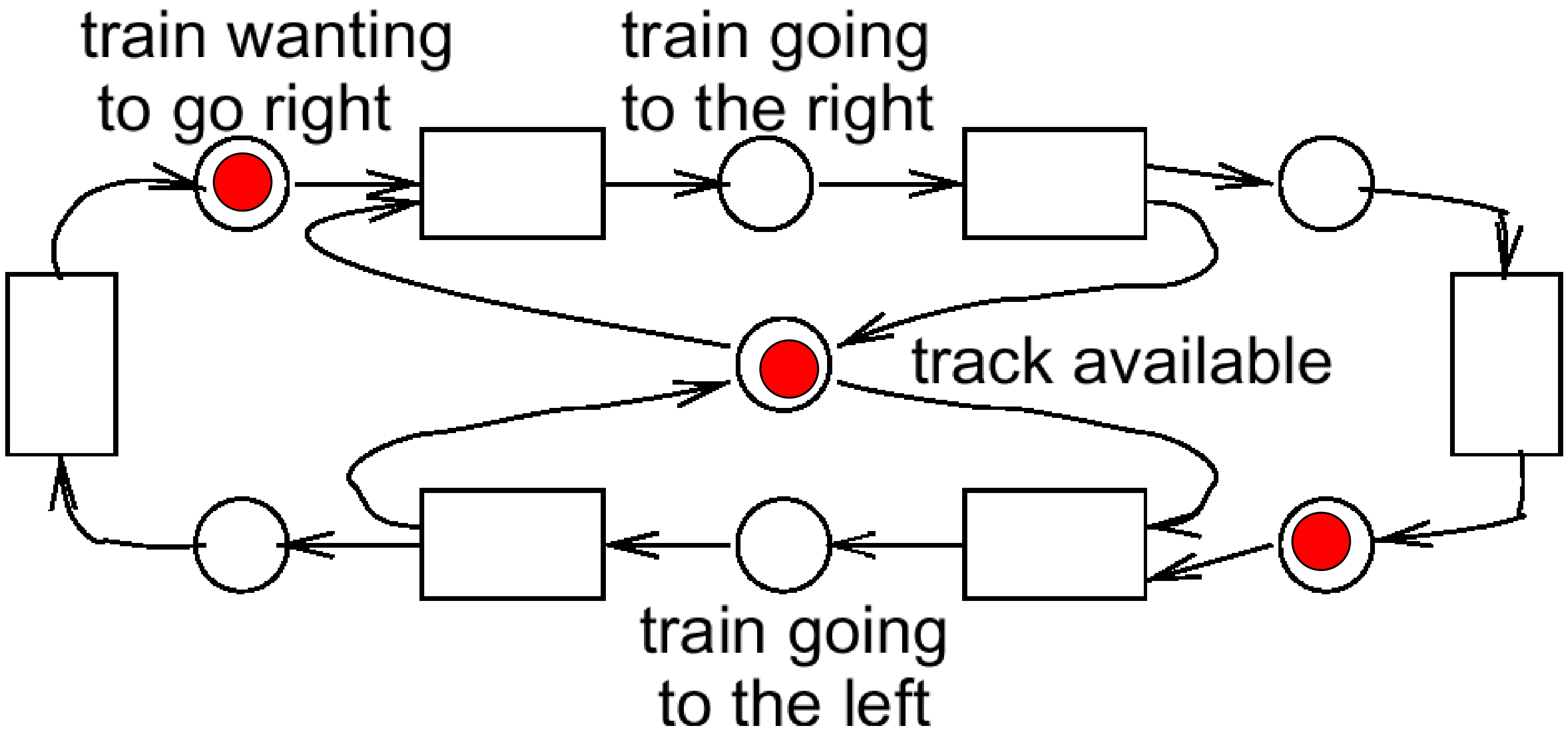
- concurrency of transitions
- alternative or conflict
- synchronization





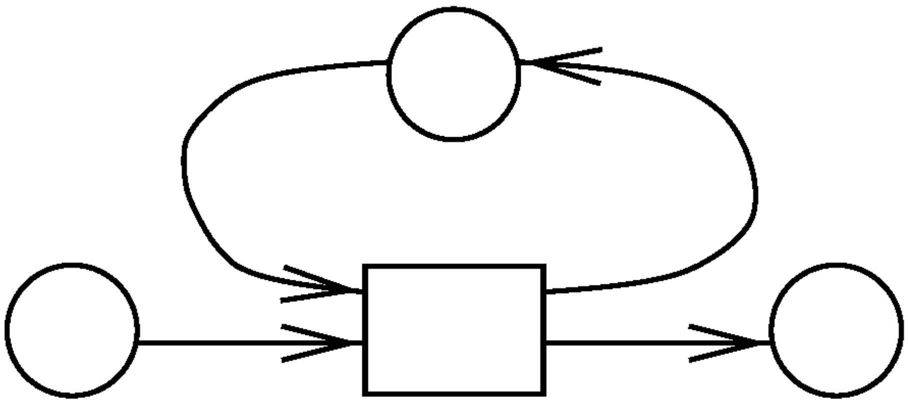


Competing Trains Example: Conflict for resource „track“



Basic structural properties: Loops and pure nets

Def.: Let $(c,e) \in C \times E$. (c,e) is called a **loop** iff $cFe \wedge eFc$.



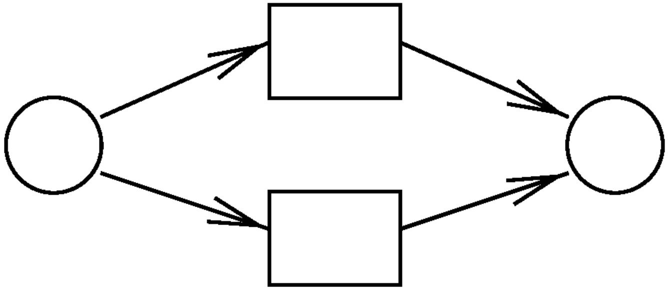
Def.: Net $N=(C,E,F)$ is called **pure**, if F does not contain any loops.

Structural properties: Simple nets

Def.: A net is called **simple**, iff

$$[x,y \in (C \cup E) \wedge ({}^{\bullet}x = {}^{\bullet}y) \wedge (x^{\bullet} = y^{\bullet})] \rightarrow x = y$$

Example (not a simple net):

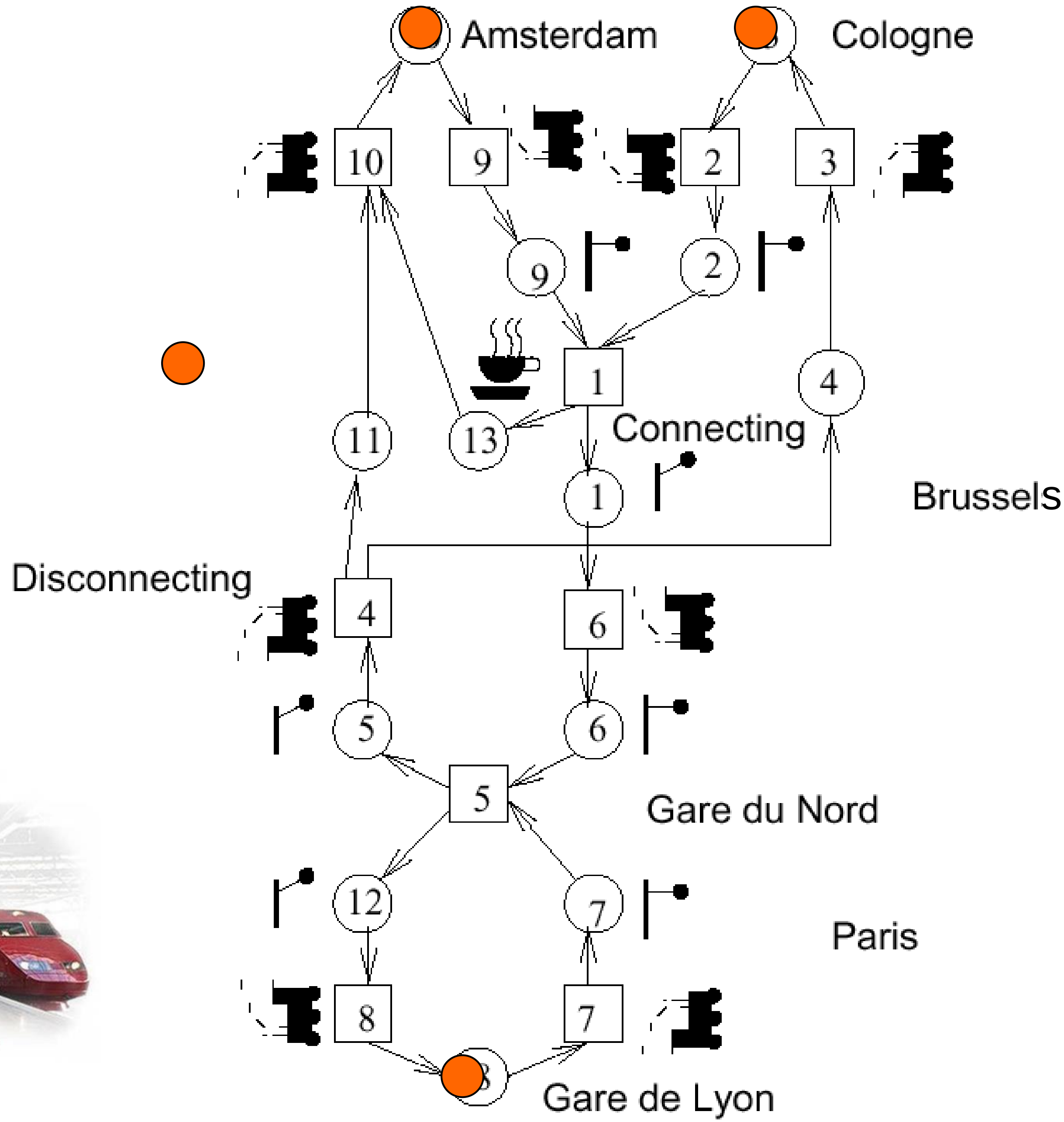


Properties of C/E

Def.:

- Marking M’ is **reachable** from marking M, iff there exists sequence of firing steps transforming M into M’ (Not.: M [*> M’)
- A C/E net is **cyclic**, iff any two different markings are reachable from each other.
- A C/E net fulfills **liveness,** iff for each marking M and for each event e there exists a reachable marking M’ that activates e for firing

Thalys trains example



Place/transition nets

- More than one token per condition, capacities of places
- weights of edges