Petri nets

Peter Marwedel
Informatik 12
TU Dortmund
Germany

Introduction

Introduced in 1962 by Carl Adam Petri in his PhD thesis.

Focus on modeling causal dependencies;
no global synchronization assumed (message passing only).

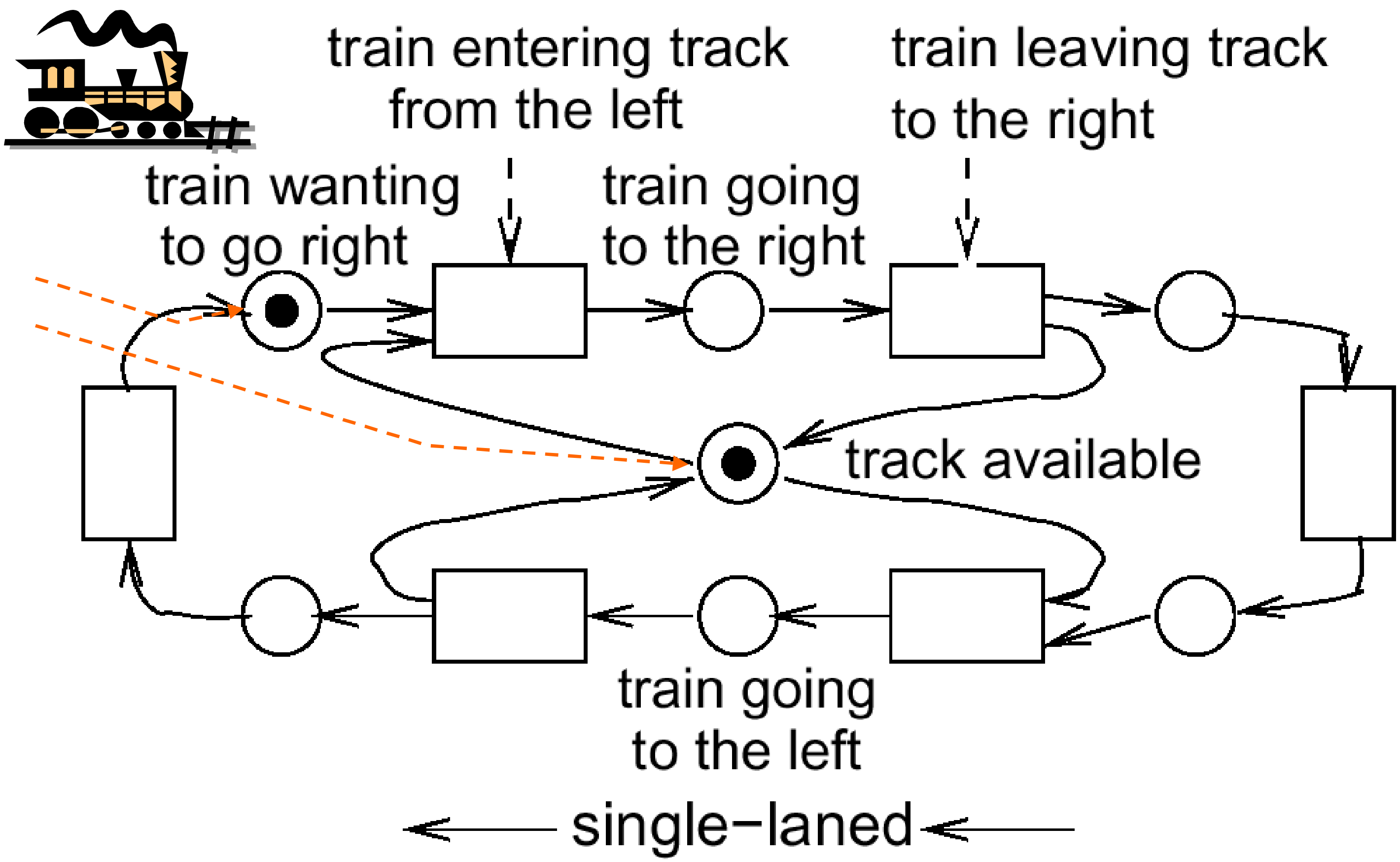
Key elements:

- **Conditions**
  Either met or no met.
- **Events**
  May take place if certain conditions are met.
- **Flow relation**
  Relates conditions and events.

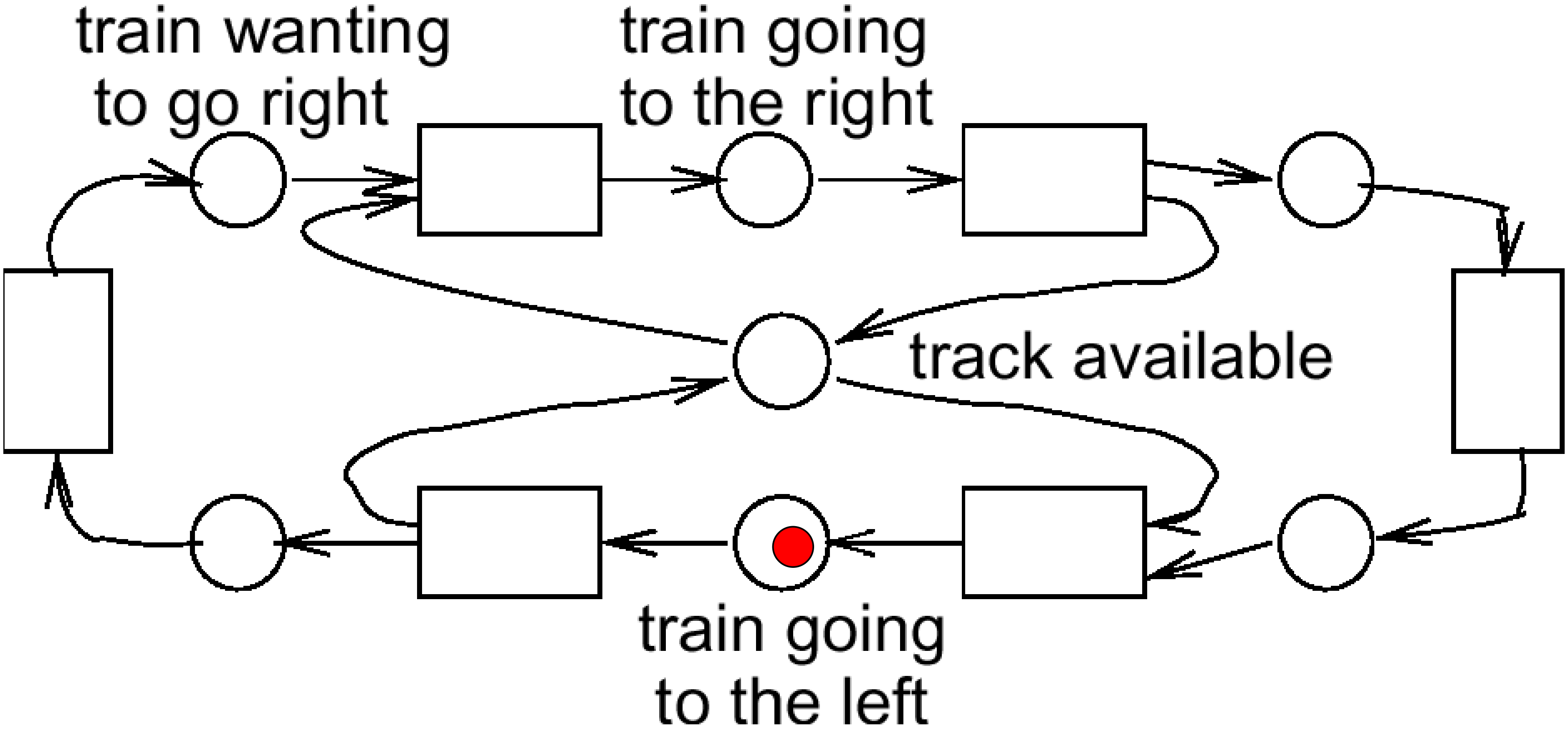
Conditions, events and the flow relation form
a **bipartite graph** (graph with two kinds of nodes).

Example:
Synchronization at single track rail segment

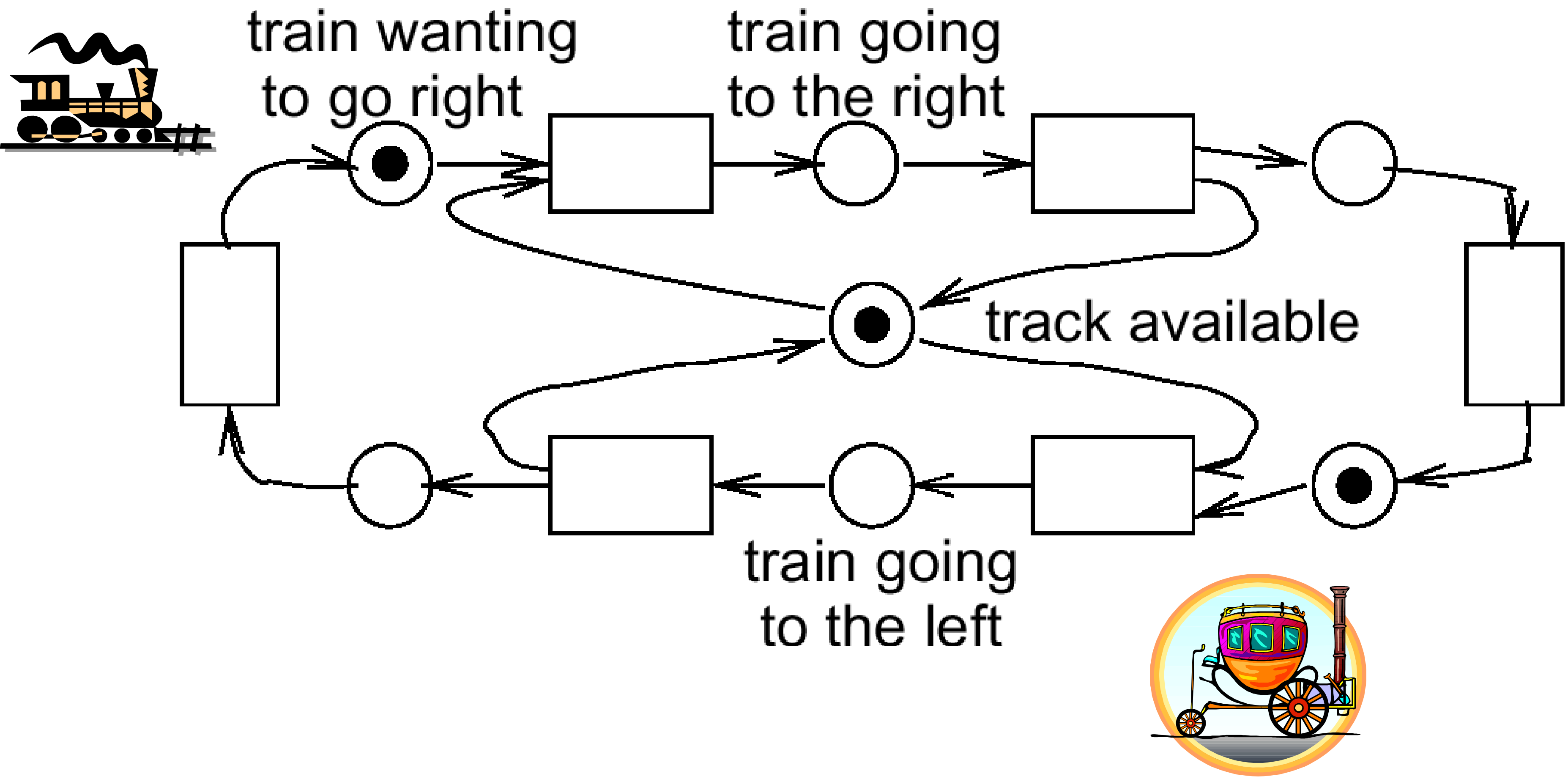
„Preconditions“



Playing the „token game"



Conflict for resource „track“



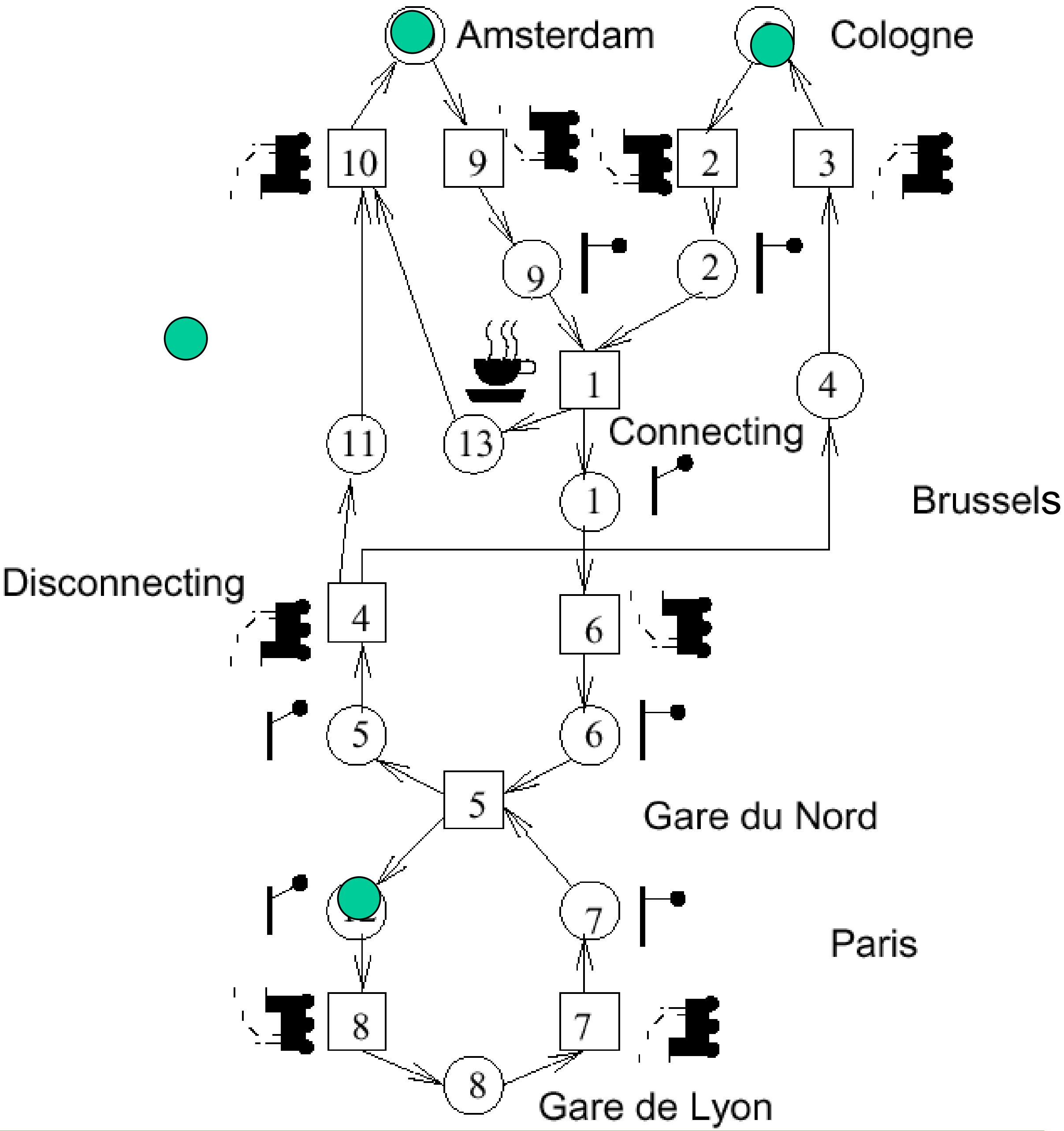
More complex example (1)

Thalys trains between Cologne, Amsterdam, Brussels and Paris.

[http://www.thalys.com/be/en]

More complex example (2)

Slightly simplified:
Synchronization at Brussels and Paris,
using stations "Gare du Nord" and "Gare de Lyon" at Paris



Condition/event nets

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

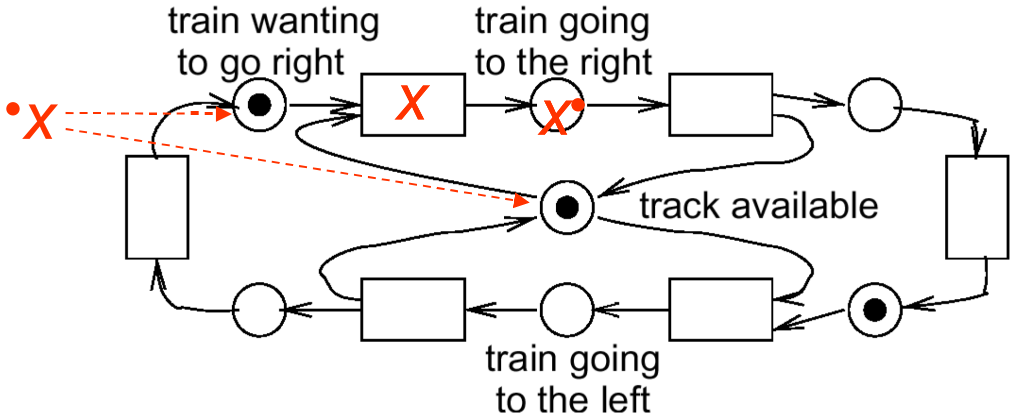
2. C and E are disjoint sets
3. $F \subseteq (C \times E) \cup (E \times C)$; is binary relation, („**flow relation**“)

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.**

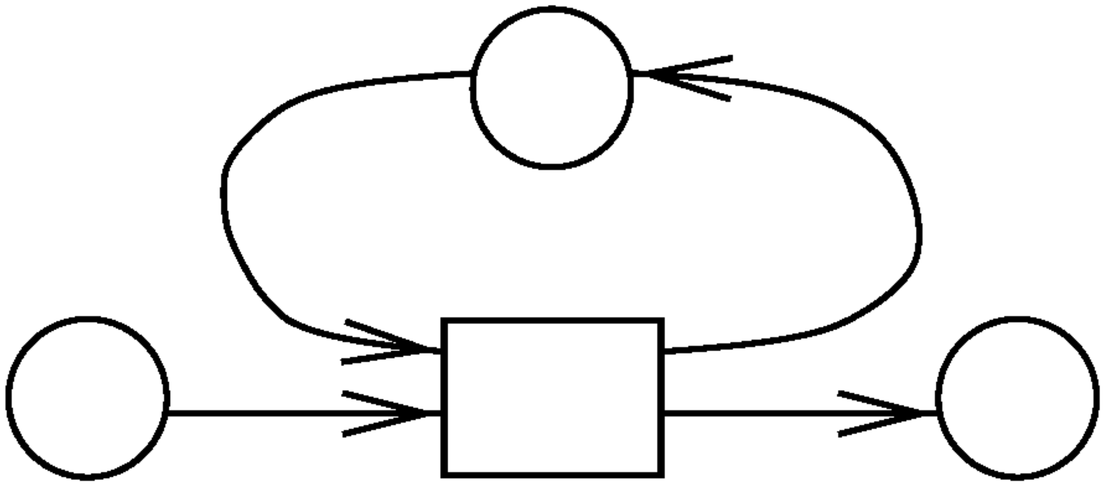
$x^{\bullet} := \{y \mid x\, F\, y\}$ is called the set of **postconditions.**

Example:



Loops and pure nets

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

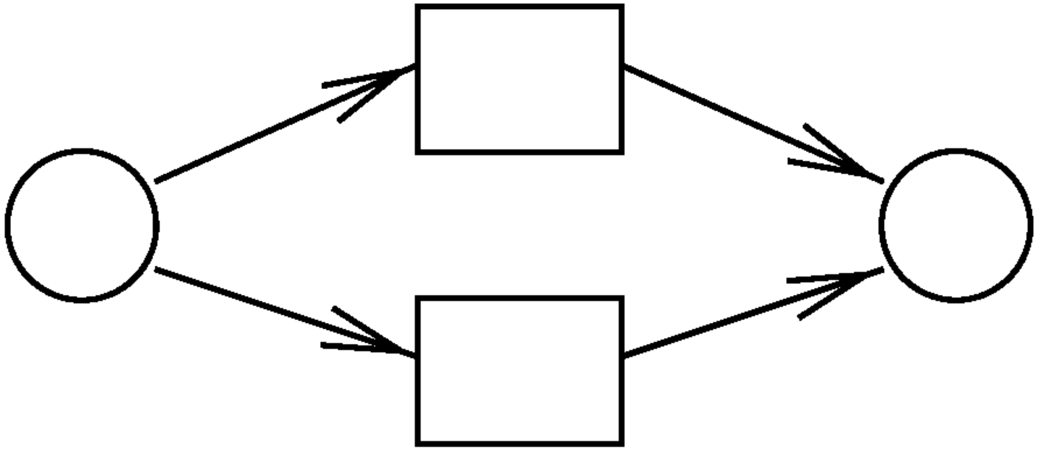


Def.: Net $N=(C,E,F)$ is called **pure**, if F does not contain any loops.

Simple nets

Def.: A net is called **simple** if no two transitions *t1* and *t2* have the same sets of input and output places.

Example (not a simple net):

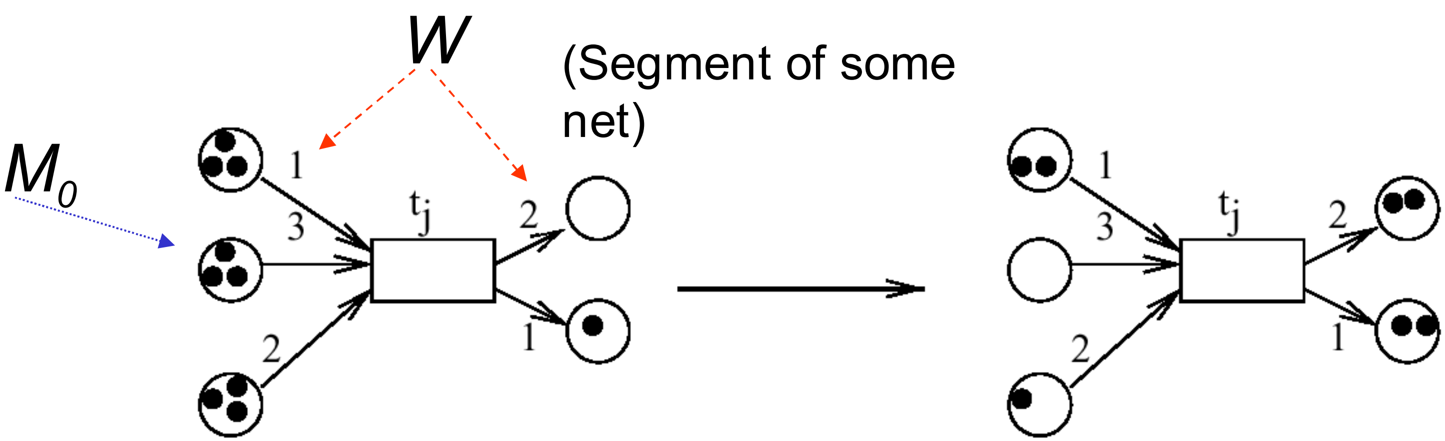


Def.: Simple nets with no isolated elements meeting some additional restrictions are called **condition/event nets (C/E nets)**.

Place/transition nets

Def.: (P, T, F, K, W, M_0) is called a **place/transition net** iff

2. $N=(P,T,F)$ is a **net** with places $p \in P$ and transitions $t \in T$
3. $K: P \rightarrow (\mathbb{N}_0 \cup \{\omega\}) \setminus \{0\}$ denotes the **capacity** of places (ω symbolizes infinite capacity)
4. $W: F \rightarrow (\mathbb{N}_0 \setminus \{0\})$ denotes the **weight of graph edges**
5. $M_0: P \rightarrow \mathbb{N}_0 \cup \{\omega\}$ represents the **initial marking** of places

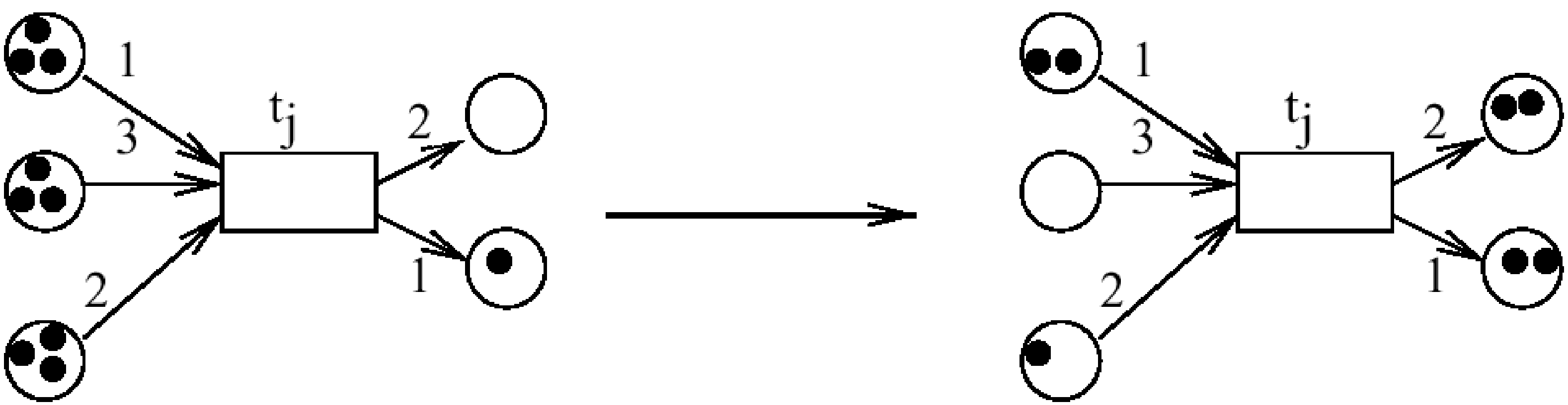


defaults:
K = ω
W = 1

Computing changes of markings

„Firing“ transitions *t* generate new markings on each of the places *p* according to the following rules:

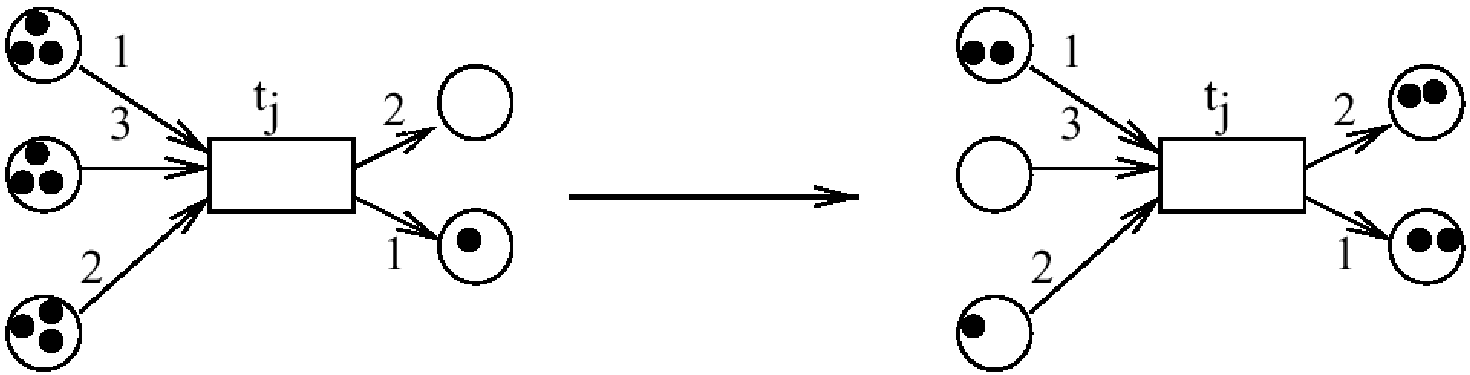
$$M'(p) = \begin{cases} M(p) - W(p,t), & \text{if } p \in {}^{\bullet}t \setminus t^{\bullet} \\ M(p) + W(t,p), & \text{if } p \in t^{\bullet} \setminus {}^{\bullet}t \\ M(p) - W(p,t) + W(t,p), & \text{if } p \in {}^{\bullet}t \cap t^{\bullet} \\ M(p) & \text{otherwise} \end{cases}$$



Activated transitions

Transition t is „activated“ iff

$$(\forall p \in {}^{\bullet}t : M(p) \geq W(p,t)) \land (\forall p \in t^{\bullet} : M(p) + W(t,p) \leq K(p))$$



Activated transitions can „take place“ or „fire“, but don‘t have to.

We never talk about „time“ in the context of Petri nets.

The order in which activated transitions fire, is not fixed (it is non-deterministic).

Applications

- Modeling of resources;
- modeling of mutual exclusion;
- modeling of synchronization.

Predicate/transition nets

Goal: compact representation of complex systems.

Key changes:

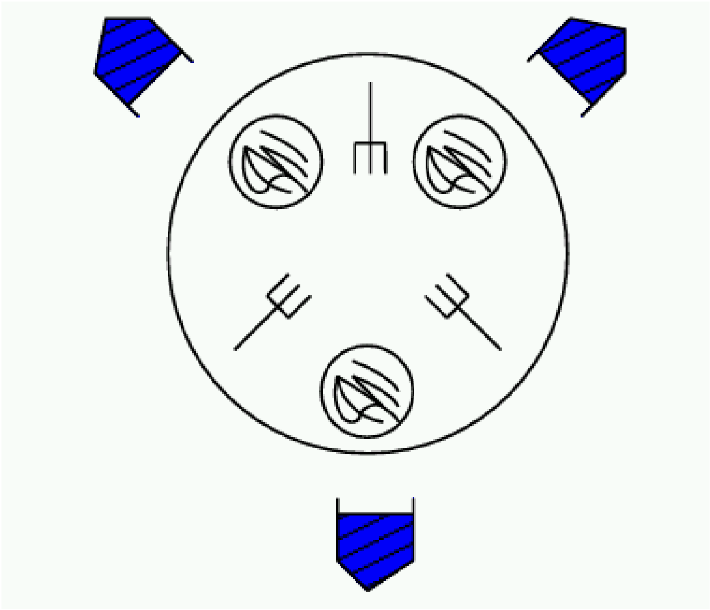
- Tokens are becoming individuals;
- Transitions enabled if functions at incoming edges true;
- Individuals generated by firing transitions defined through functions

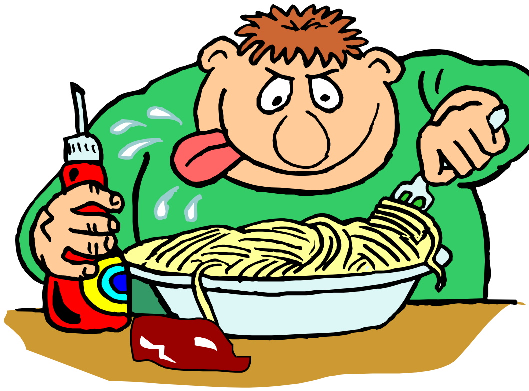
Changes can be explained by folding and unfolding C/E nets,

☞ semantics can be defined by C/E nets.

Example: Dining philosophers problem

n>1 philosophers sitting at a round table;
n forks,
n plates with spaghetti;
philosophers either thinking or eating spaghetti
(using left and right fork).





2 forks needed!

How to model conflict for forks?

How to guarantee avoiding starvation?

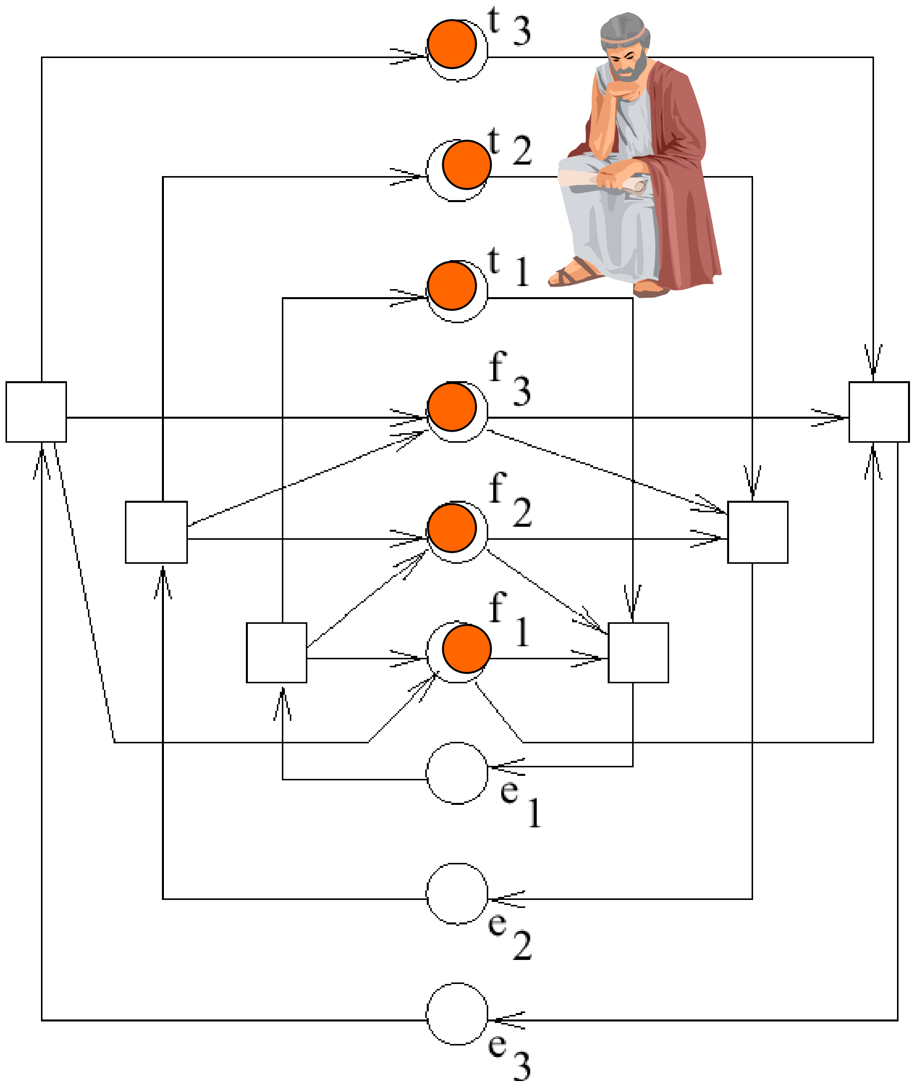
Condition/event net model of the dining philosophers problem

Let x ∈ {1..3}

t_x: x is thinking

e_x: x is eating

f_x: fork x is available

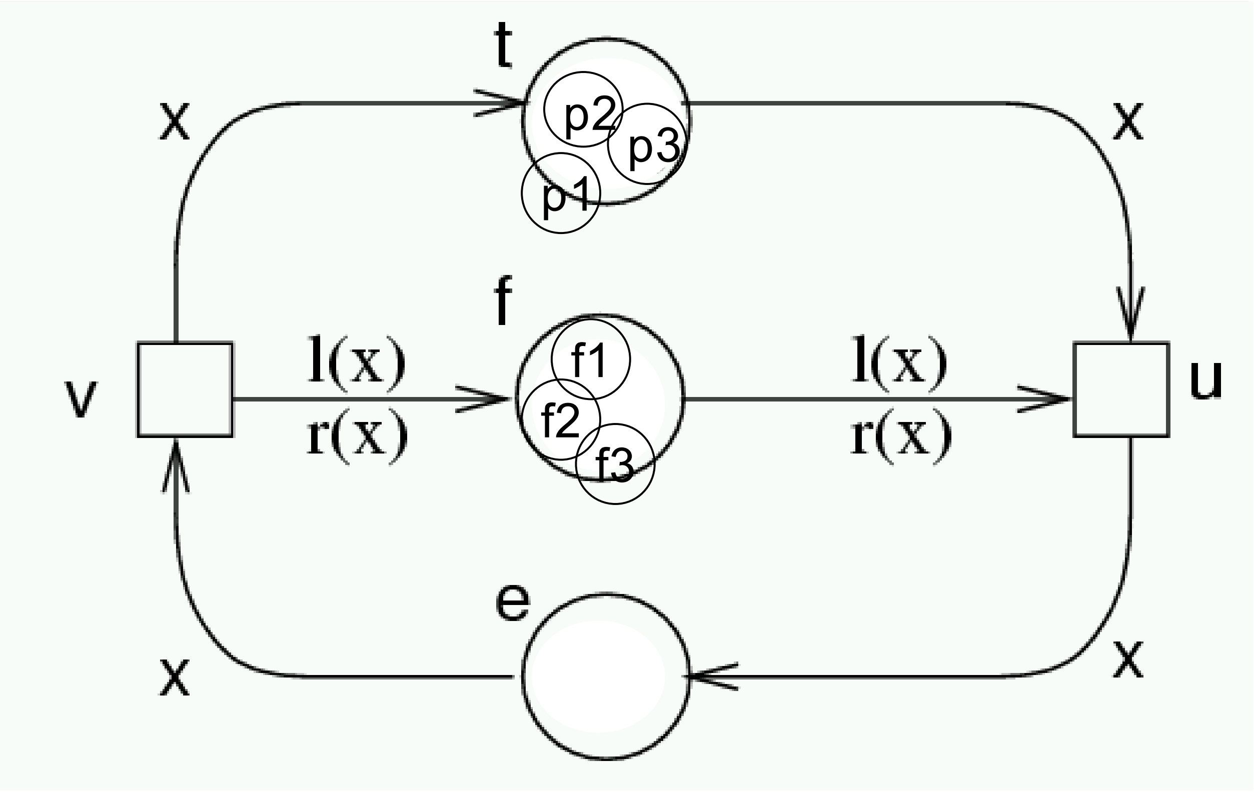


Model quite clumsy.

Difficult to extend to more philosophers.

Predicate/transition model of the dining philosophers problem (1)

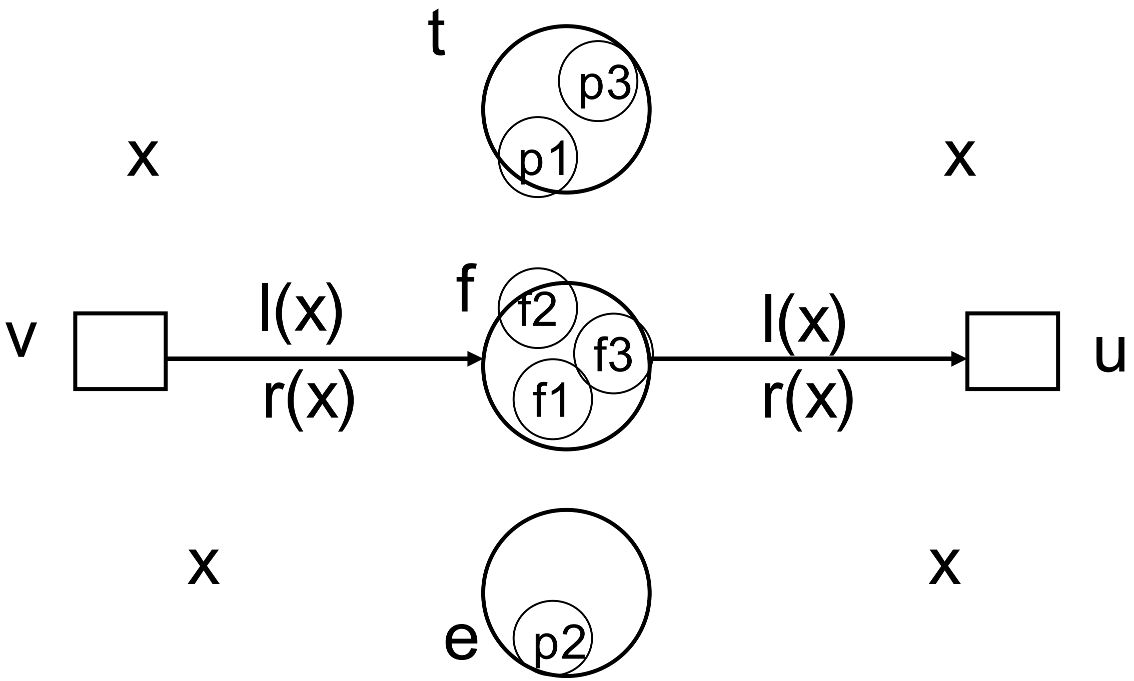
Let x be one of the philosophers,
let l(x) be the left spoon of x,
let r(x) be the right spoon of x.



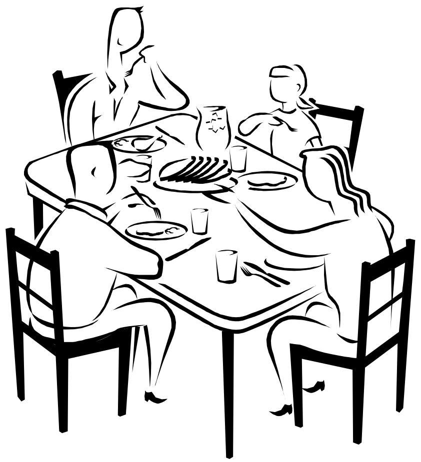
Tokens: individuals.

Semantics can be defined by replacing net by equivalent condition/event net.

Predicate/transition model of the dining philosophers problem (2)



Model can be extended to arbitrary numbers of people.



Evaluation

Pros:

- Appropriate for distributed applications,
- Well-known theory for formally proving properties,
- Initially a quite bizarre topic, but now accepted due to increasing number of distributed applications.

Cons (for the nets presented) :

- problems with modeling timing,
- no programming elements,
- no hierarchy.

Extensions:

- Enormous amounts of efforts on removing limitations.

Summary

Petri nets: focus on causal relationships

Condition/event nets

- Single token per place

Place/transition nets

- Multiple tokens per place

Predicate/transition nets

- Tokens become individuals
- Dining philosophers used as an example

Extensions required to get around limitations