ECSE 421 Lecture 5: Petri Nets

ESD Chapter 2

Last Time

- Kahn Process Networks
- Synchronous Data Flow
- Simulink

Where Are We?

W	D	Date		Topic	ESD	PES	Out	In	Notes
1	T	12-Jan-2016	L01	Introduction to Embedded System Design	1.1-1.4				
	R	14-Jan-2016		Introduction to Embedded System Design	1.1-1.4				
2	T	19-Jan-2016	L02	Specifying Requirements / MoCs / MSC	2.1-2.3				
	R	21-Jan-2016	L03	CFSMs	2.4				
3	T	26-Jan-2016	L04	Data Flow Modeling	2.5	3.1-5,7	LA1		
	R	28-Jan-2016	L05	Petri Nets	2.6				
4	T	2-Feb-2016	L06	Discrete Event Models	2.7	4			Guest lecturer
	R	4-Feb-2016	L07	DES / Von Neumann Model of Computation	2.8-2.10	5	LA2	LA1	
5	T	9-Feb-2016	L08	Sensors	3.1-3.2	7.3,12.1-6			
	R	11-Feb-2016	L09	Processing Elements	3.3	12.6-12			
6	T	16-Feb-2016	L10	More Processing Elements / FPGAs			LA3	LA2	
	R	18-Feb-2016	L11	Memories, Communication, Output	3.4-3.6				
7	T	23-Feb-2016	L12	Embedded Operating Systems	4.1				
	R	25-Feb-2016		***Midterm exam: in-class, closed book***			P	LA3	Chapters 1-3
	T	1-Mar-2016		***No class***					Winter break
	R	3-Mar-2016		***No class***					Winter break
8	T	8-Mar-2016	L13	Middleware	4.4-4.5				
	R	10-Mar-2016	L14	Performance Evaluation	5.1-5.2				
9	T	15-Mar-2016	L15	More Evaluation and Validation	5.3-5.8				
	R	17-Mar-2016	L16	Introduction to Scheduling	6.1-6.2.2				
10	T	22-Mar-2016	L17	Scheduling Aperiodic Tasks	6.2.3-6.2.4				
	R	24-Mar-2016	L18	Scheduling Periodic Tasks	6.2.5-6.2.6				

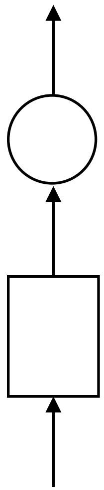
McGill

MoCs Considered in 421

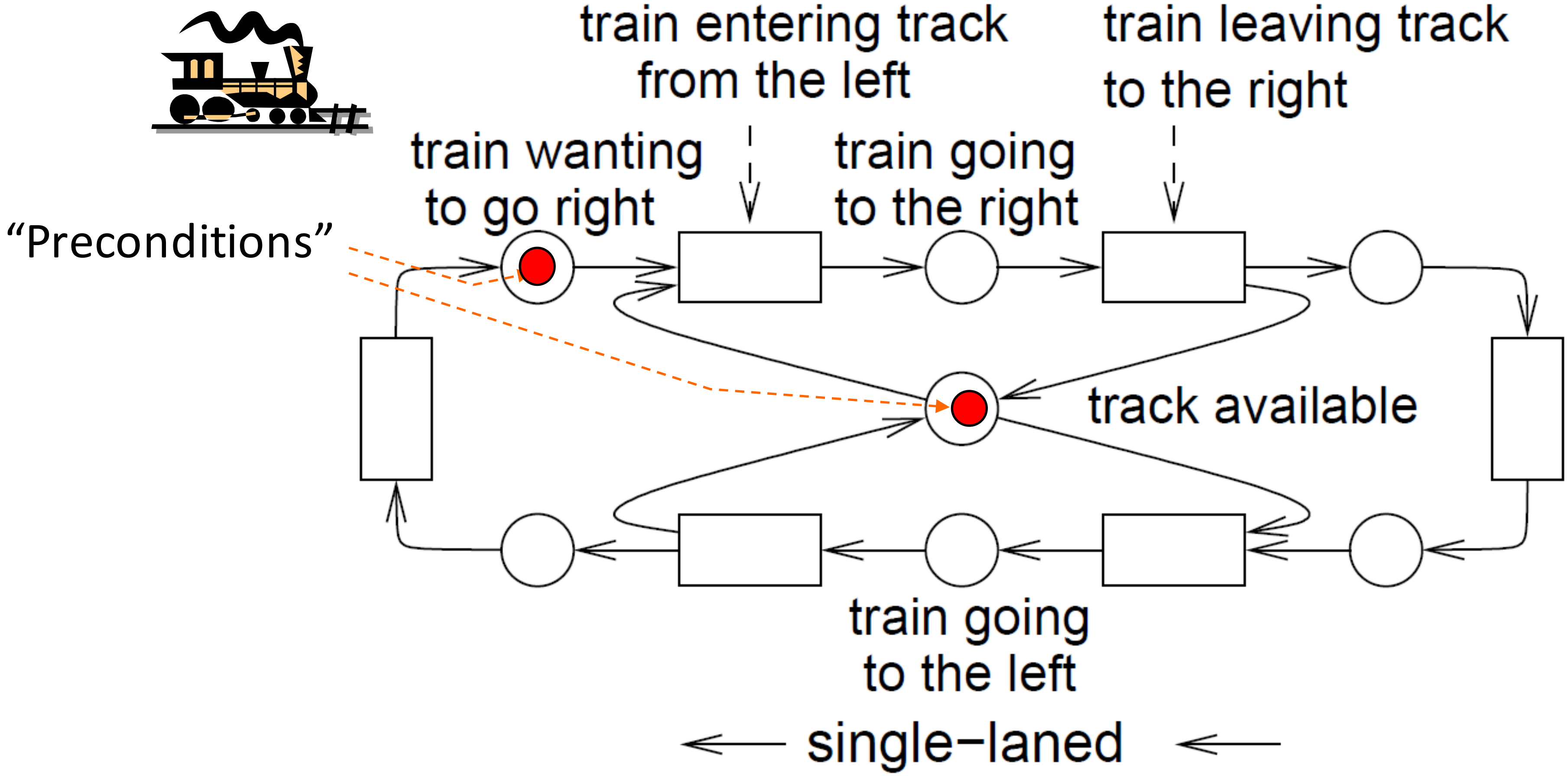
Communication/ local computations	Shared memory	Message passing Synchronous | Asynchronous	
Undefined components	Plain text, use cases		
	| (Message) sequence charts		
Communicating finite state machines	StateCharts		SDL
Data flow	(Not useful)		Kahn networks, SDF
Petri nets		**C/E nets, P/T nets, …**	
Discrete event (DE) model	VHDL*, Verilog*, SystemC*, …	Only experimental systems, e.g. distributed DE in Ptolemy	
Von Neumann model	C, C++, Java	C, C++, Java with libraries	
		CSP, ADA |	

Introduction to Petri Nets

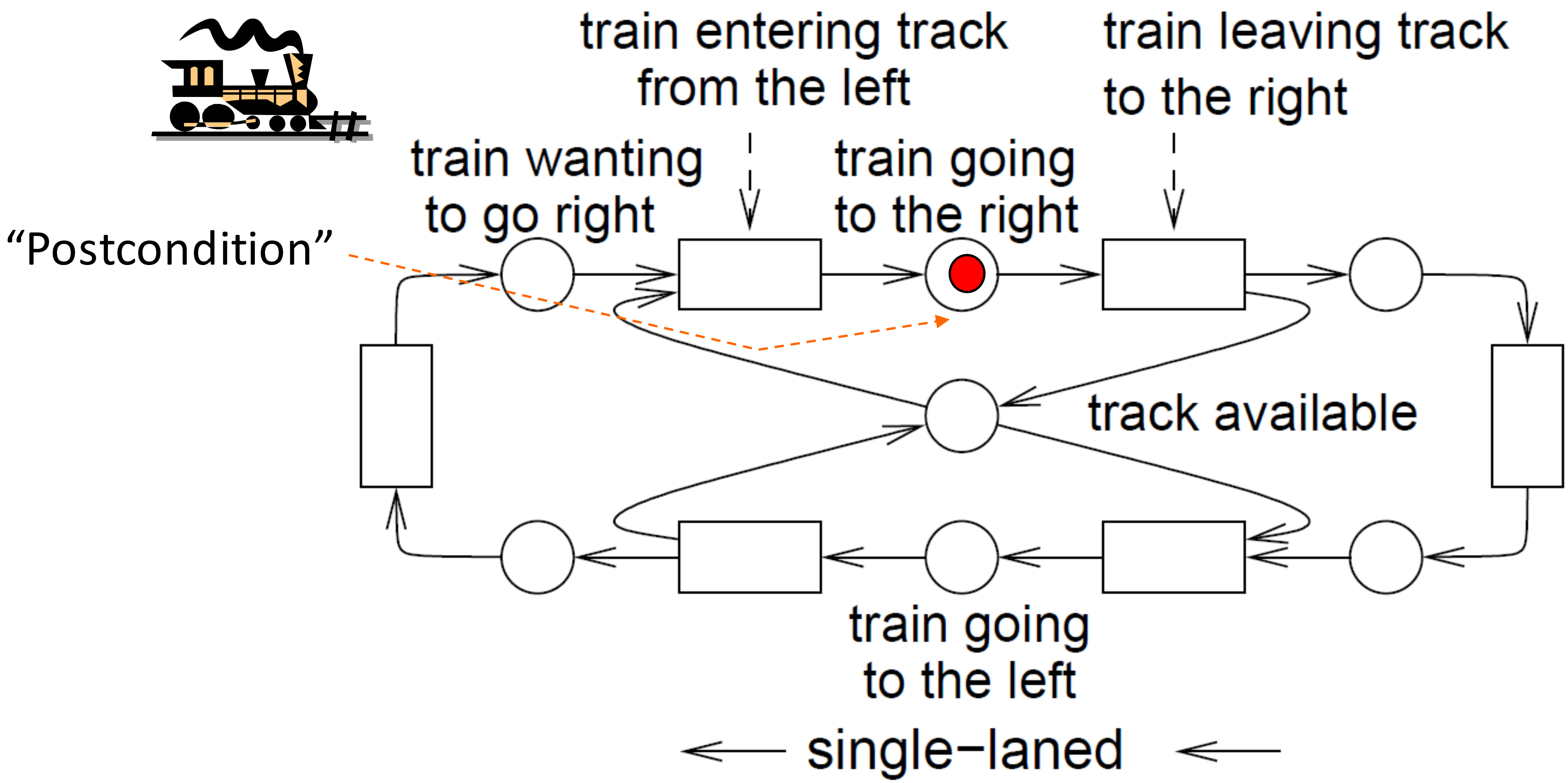
- Introduced in 1962 by Carl Adam Petri
- Model causal dependencies
- Communication via message passing only
- Key elements:
  - Conditions: either met or not met
  - Events: may take place if conditions are met
  - Flow relation: relates conditions and events
- Conditions, events and the flow relation form a bipartite graph (graph with two kinds of nodes)



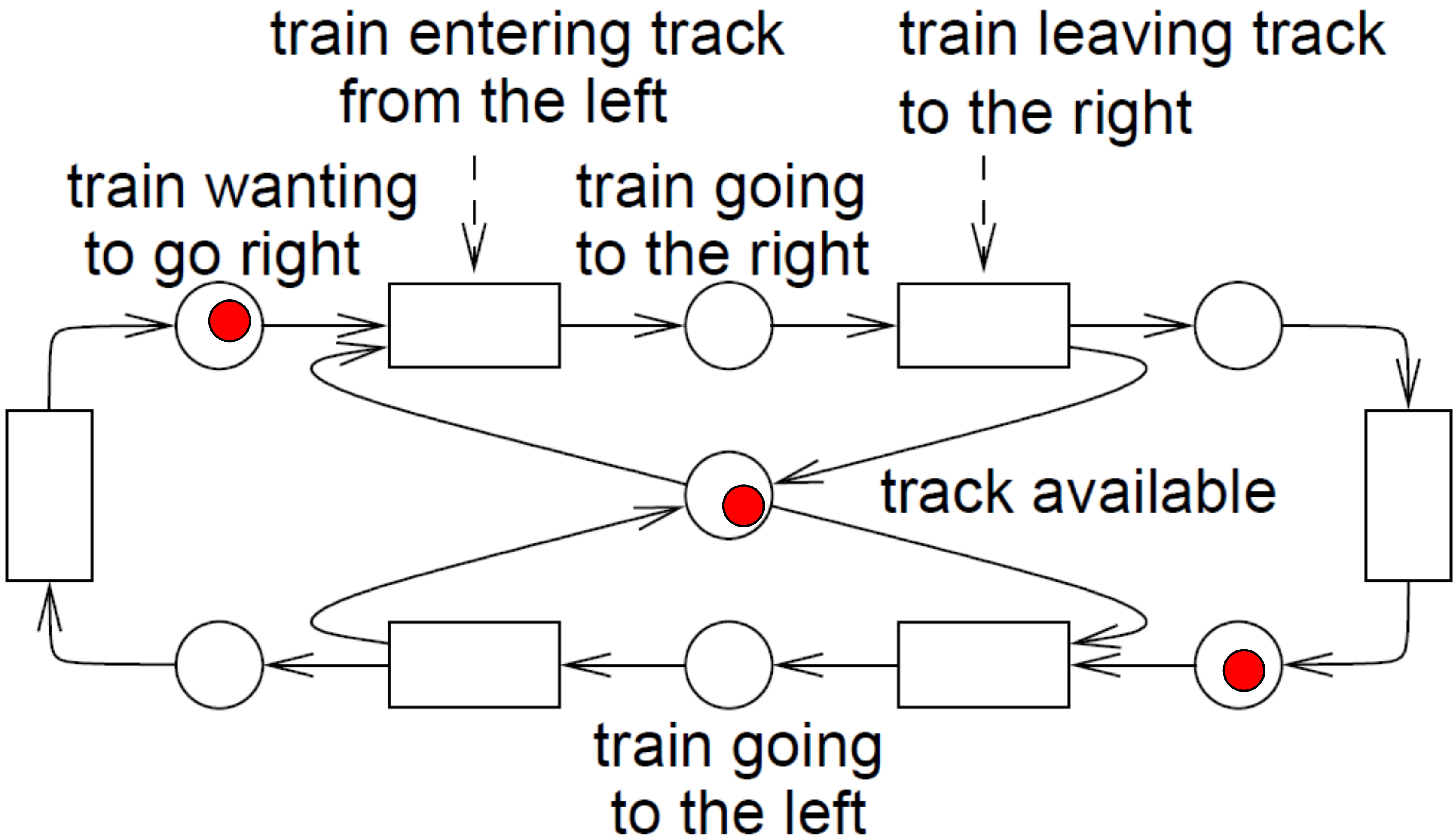
Example: Rail Segment Synchronization



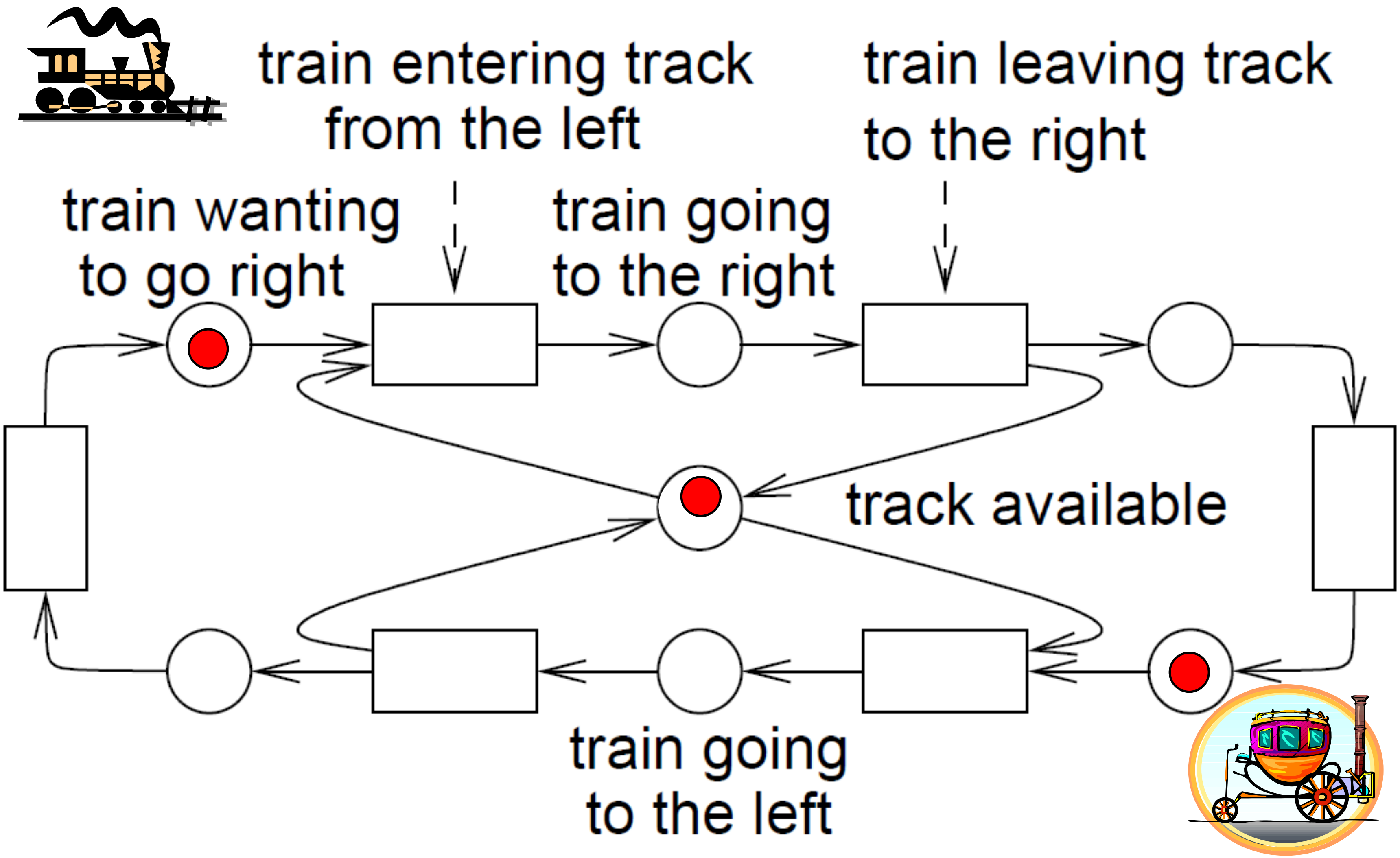
Example: Rail Segment Synchronization



Playing the "Token Game"



Conflict for Resource "Track"

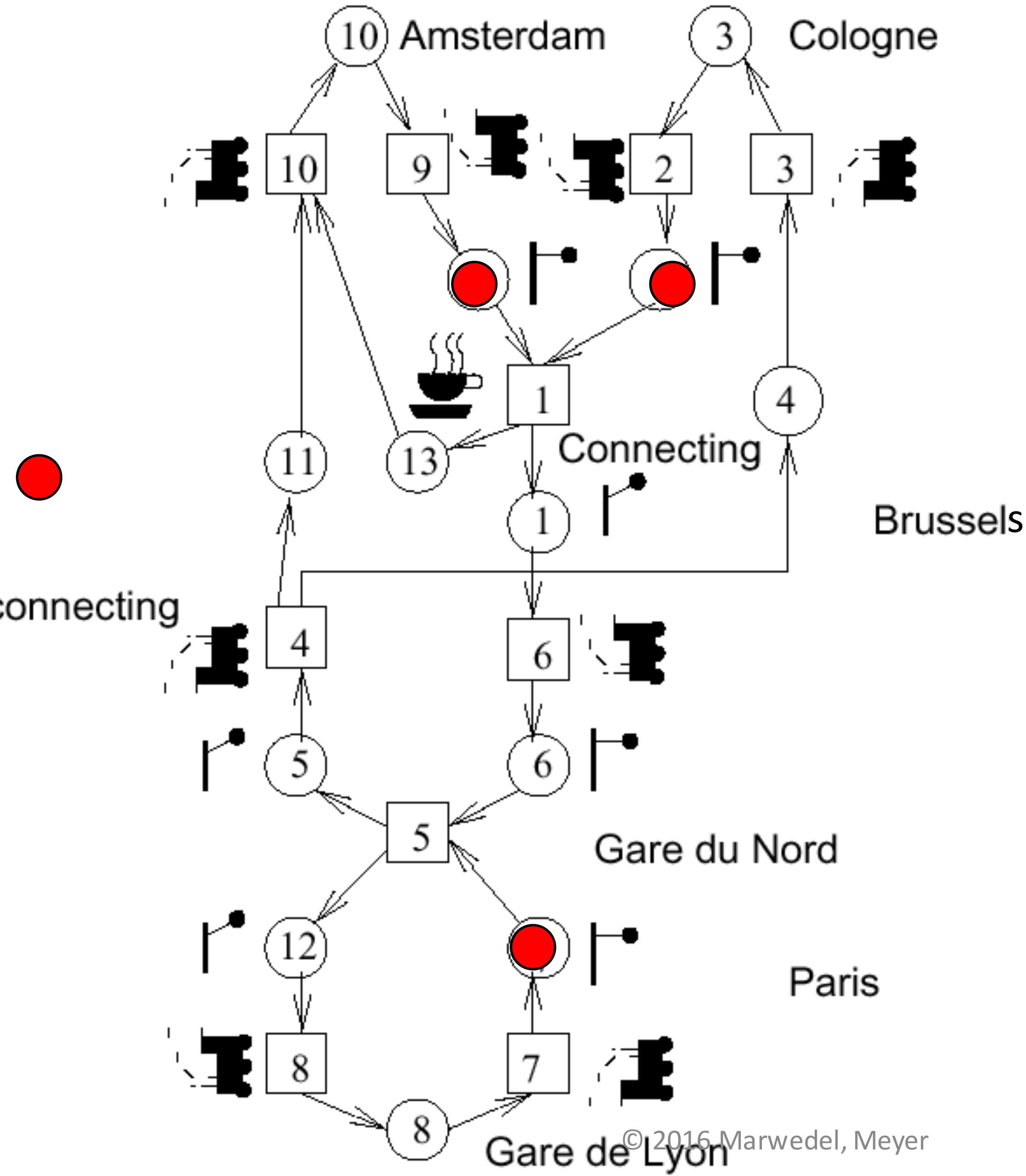


A More Complex Example

- Thalys trains between Cologne, Amsterdam, Brussels and Paris

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

Thalys Trains Synchronization



- Slightly simplified
- Synchronization of trains at Brussels
- Synchronization of trains at “Gare du Nord” in Paris

Condition/Event Nets

Def.: $N=(C,E,F)$ is called a **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")

- What does this mean?
  – Vertices can't be both conditions and events
  – Edges must only connect conditions to events (& vv)

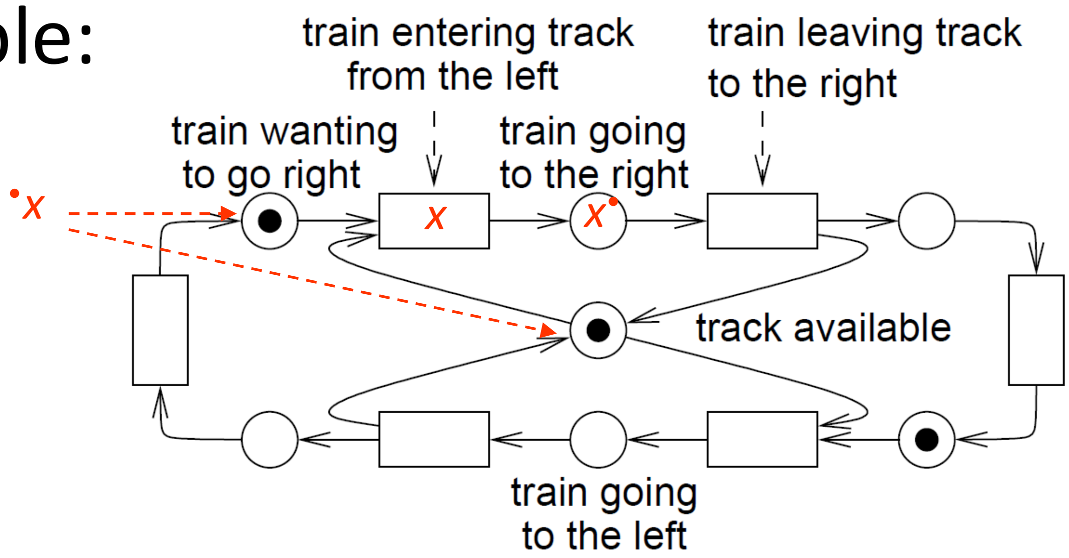
Pre- and Post-sets

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

$\bullet x := \{y \mid y\, F\, x\}$ is called the **pre-set** of x,
(or **preconditions** if $x \in E$)

$x \bullet := \{y \mid x\, F\, y\}$ is called the set of **post-set** of x,
(or **postconditions** if $x \in E$)

- Example:

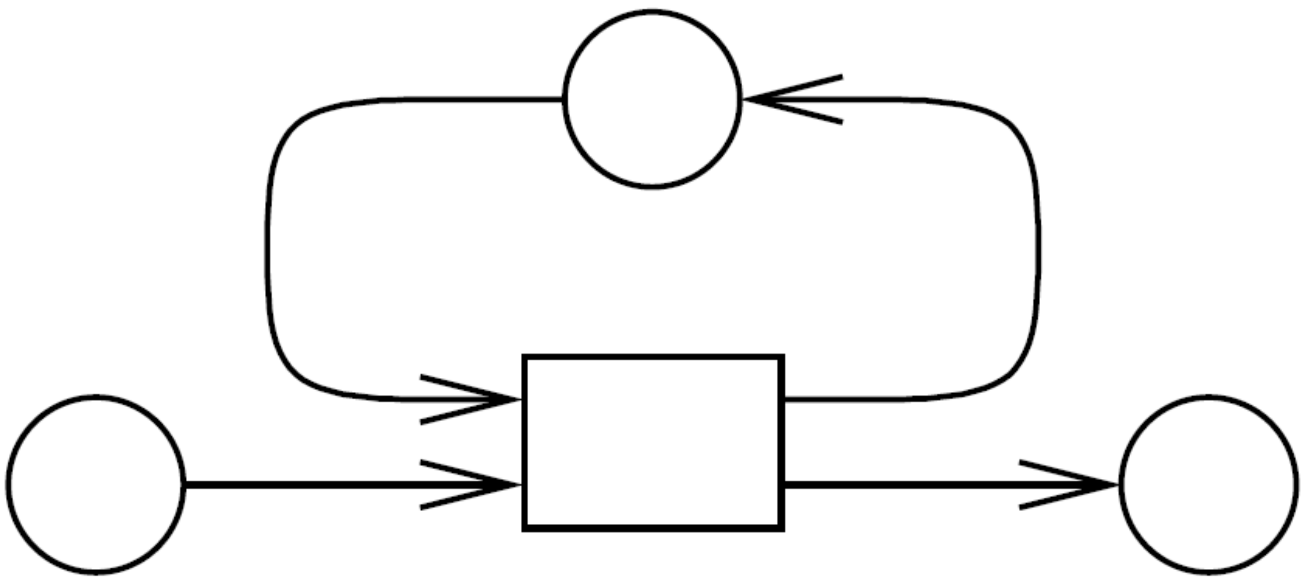


Loops and Pure Nets

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

$cFe \wedge eFc$.

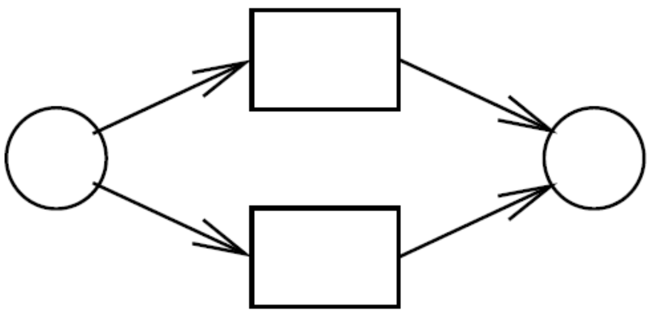
Def.: Net $N=(C,E,F)$ is called **pure** if F does not contain any loops.

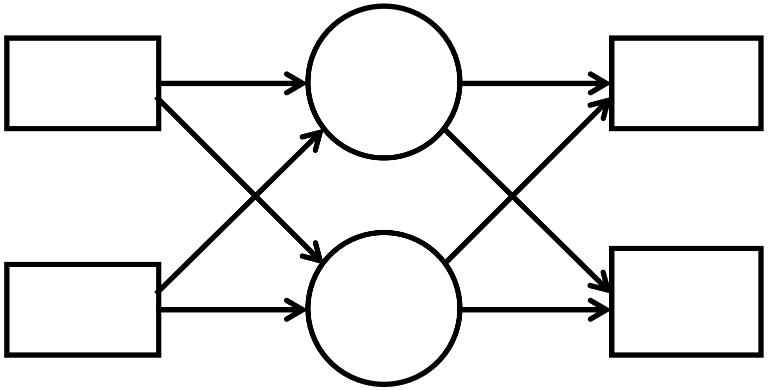


Simple Nets

Def.: A net is called **simple** if no two nodes n_1 and n_2 have the same pre-set and post-set.

- Example (*not* simple nets):





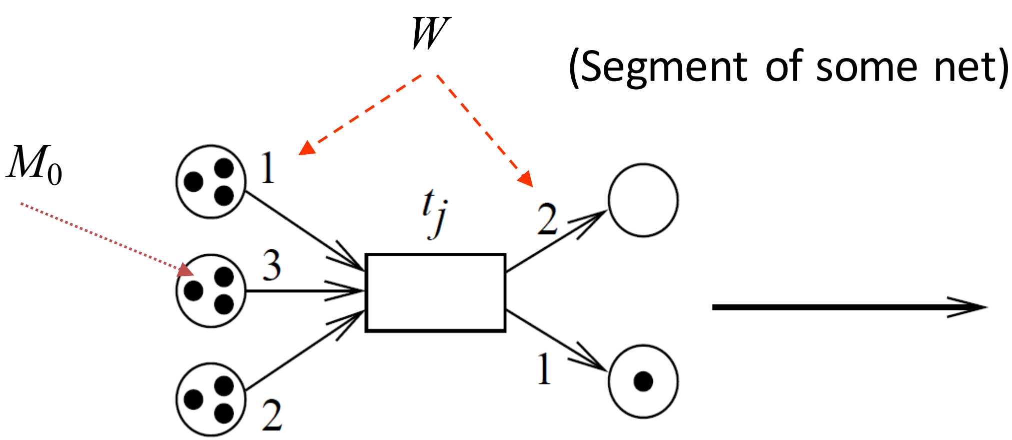
Def.: Simple nets with no isolated elements meeting some additional restrictions are called **condition/event nets** or **C/E nets**.

Place/Transition Nets

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

1. $N=(P, T, F)$ is a **net** with places $p \in P$ and transitions $t \in T$
2. $K: P \rightarrow (\mathbb{N}_0 \cup \{\omega\}) \setminus \{0\}$ denotes the **capacity** of places

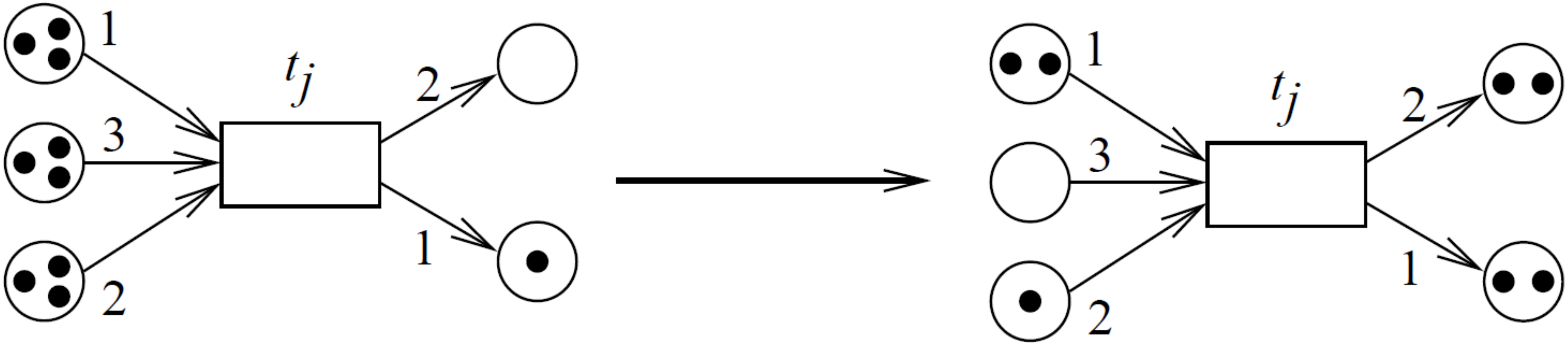
   (ω symbolizes infinite capacity)
3. $W: F \rightarrow (\mathbb{N}_0 \setminus \{0\})$ denotes the **weight of graph edges**
4. $M_0: P \rightarrow \mathbb{N}_0 \cup \{\omega\}$ represents the **initial marking** of places



Computing Changes of Markings

- "Firing" transitions *t* generate new markings on each of the places *p* according to the following:

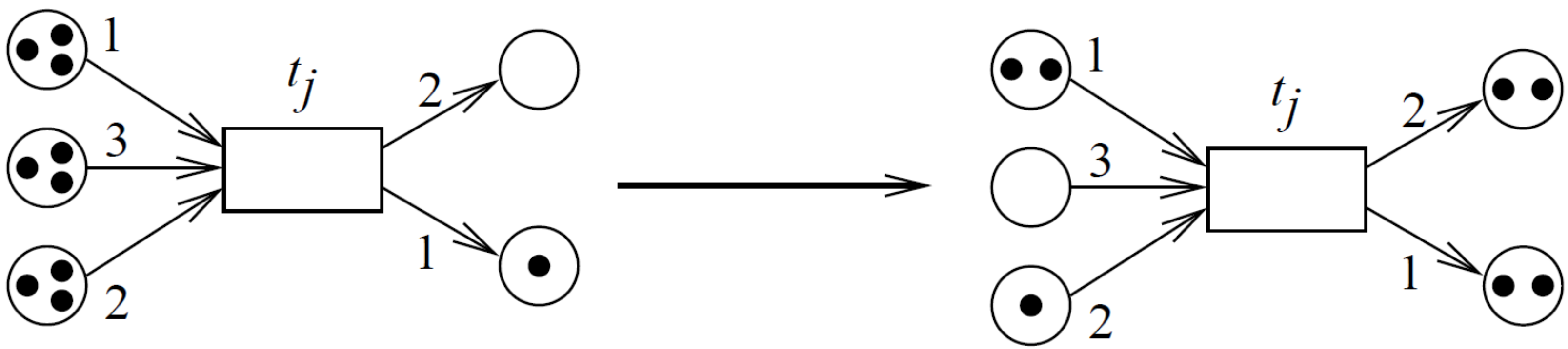
$$M'(p) = \begin{cases} M(p) - W(p,t), & \text{if } p \in {}^{\bullet}t \setminus t^{\bullet} \\ M(p) + W(t,p), & \text{if } p \in t^{\bullet} \setminus {}^{\bullet}t \\ M(p) - W(p,t) + W(t,p), & \text{if } p \in {}^{\bullet}t \cap t^{\bullet} \\ M(p) & \text{otherwise} \end{cases}$$



Activated Transitions

- Transition *t* is "activated" iff

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



- Activated transitions can fire but don't have to
- No notion of "time" in Petri nets
- The order in which activated transitions fire is not fixed (it is non-deterministic)

Shorthand for Changes of Markings

Slide 16:

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

Let

$$\underline{t}(p) = \begin{cases} -W(p,t) \text{ if } p \in {}^{\bullet}t \setminus t^{\bullet} \\ +W(t,p) \text{ if } p \in t^{\bullet} \setminus {}^{\bullet}t \\ -W(p,t) + W(t,p) \text{ if } p \in t^{\bullet} \cap {}^{\bullet}t \\ 0 \end{cases}$$

$\Rightarrow$ $\forall\, p \in P\!: \; M'(p) = M(p) + \underline{t}(p)$

$\Rightarrow$ $M' = M + \underline{t}$ +: vector add

Representing Markings with Matrix $\underline{N}$

$$\underline{t}(p) = \begin{cases} -W(p,t) \text{ if } p \in {}^{\bullet}t \setminus t^{\bullet} \\ +W(t,p) \text{ if } p \in t^{\bullet} \setminus {}^{\bullet}t \\ -W(p,t) + W(t,p) \text{ if } p \in t^{\bullet} \cap {}^{\bullet}t \\ 0 \end{cases}$$

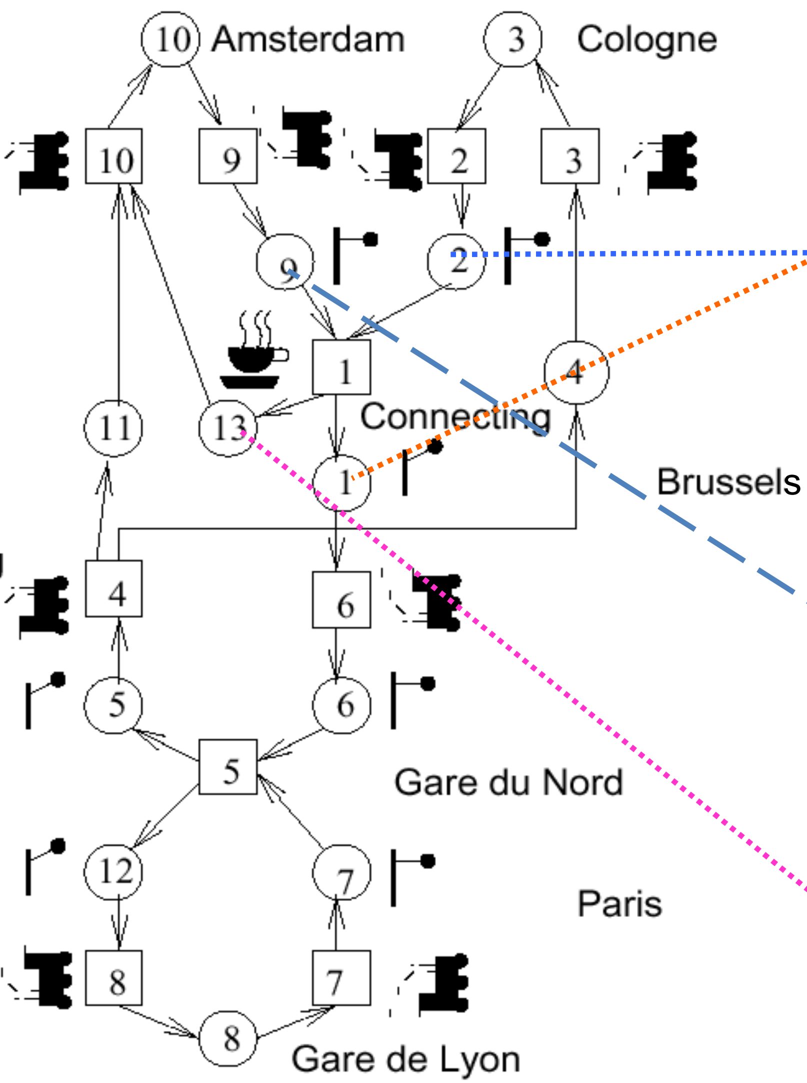
Def.: Matrix $\underline{N}$ of net N is a mapping

$\underline{N}$: $P \times T \rightarrow \mathbb{Z}$ (integers)

such that $\forall\ t \in T : \underline{N}(p,t) = \underline{t}(p)$

- Components in column t and row p indicate the change of the marking of place p if transition t takes place
- For pure nets, $(\underline{N}, M_0)$ is a complete representation of a net

Example: $\underline{N}$ =



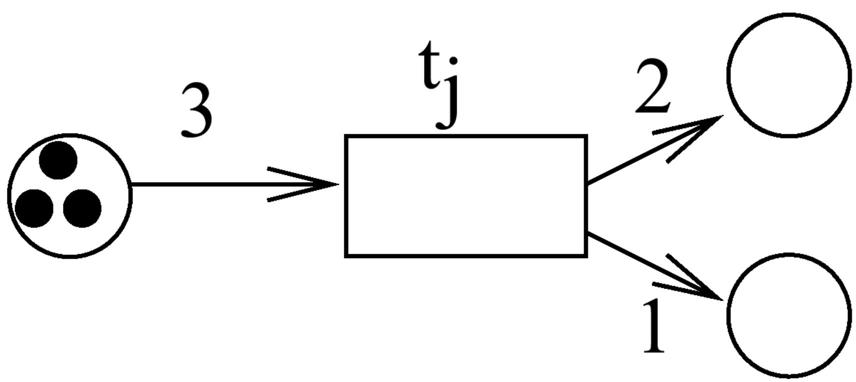
	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
p_1	1					−1				
p_2	−1	1								
p_3		−1	1							
p_4			−1	1						
p_5				−1	1					
p_6					−1	1				
p_7					−1		1			
p_8							−1	1		
p_9	−1							1	1	
p_{10}									−1	1
p_{11}				1						−1
p_{12}					1			−1		
p_{13}	1									−1

Place Invariants

- Standardized technique for proving properties of system models
- For any transition $t_j \in T$ we are looking for sets $R \subseteq P$ of places for which the accumulated marking is constant:

$$\sum_{p \in R} \underline{t}_j(p) = 0$$

- Example:



Characteristic Vector

$$\sum_{p \in R} \underline{t}_j(p) = 0$$

Let: $$\underline{c}_R(p) = \begin{cases} 1 \text{ if } p \in R \\ 0 \text{ if } p \notin R \end{cases}$$

$$\Rightarrow \quad 0 = \sum_{p \in R} \underline{t}_j(p) = \sum_{p \in P} \underline{t}_j(p)\, \underline{c}_R(p) = \underline{t}_j \cdot \underline{c}_R$$

Dot product

Condition for Place Invariants

$$\sum_{p \in R} \underline{t}_j(p) = \sum_{p \in P} \underline{t}_j(p)\, \underline{c}_R(p) = \underline{t}_j \cdot \underline{c}_R = 0$$

- The accumulated marking is constant for **all** transitions if

$$\begin{array}{ccc} \underline{t}_1 \cdot \underline{c}_R & = & 0 \\ \dots & \dots & \dots \\ \underline{t}_n \cdot \underline{c}_R & = & 0 \end{array}$$

Equivalent to $\underline{\boldsymbol{N}}^T \boldsymbol{c_R} = \mathbf{0}$ where $\underline{\boldsymbol{N}}^T$ is the transpose of $\underline{\boldsymbol{N}}$

More Detailed View of Computations

$$\begin{pmatrix} \underline{t}_1(p_1)\ldots\underline{t}_1(p_n) \\ \underline{t}_2(p_1)\ldots\underline{t}_2(p_n) \\ \ldots \\ \underline{t}_m(p_1)\ldots\underline{t}_m(p_n) \end{pmatrix} \begin{pmatrix} \underline{c}_R(p_1) \\ \underline{c}_R(p_2) \\ \ldots \\ \underline{c}_R(p_n) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

- System of linear equations
  - Solution vectors must consist of zeros and ones
- Equations with multiple unknowns that must be integers are called diophantic (Greek mathematician Diophantus, ~300 B.C.)
  - More complex to solve than standard system of linear equations (actually NP-hard)
- Different techniques for solving equation system (manual, …)

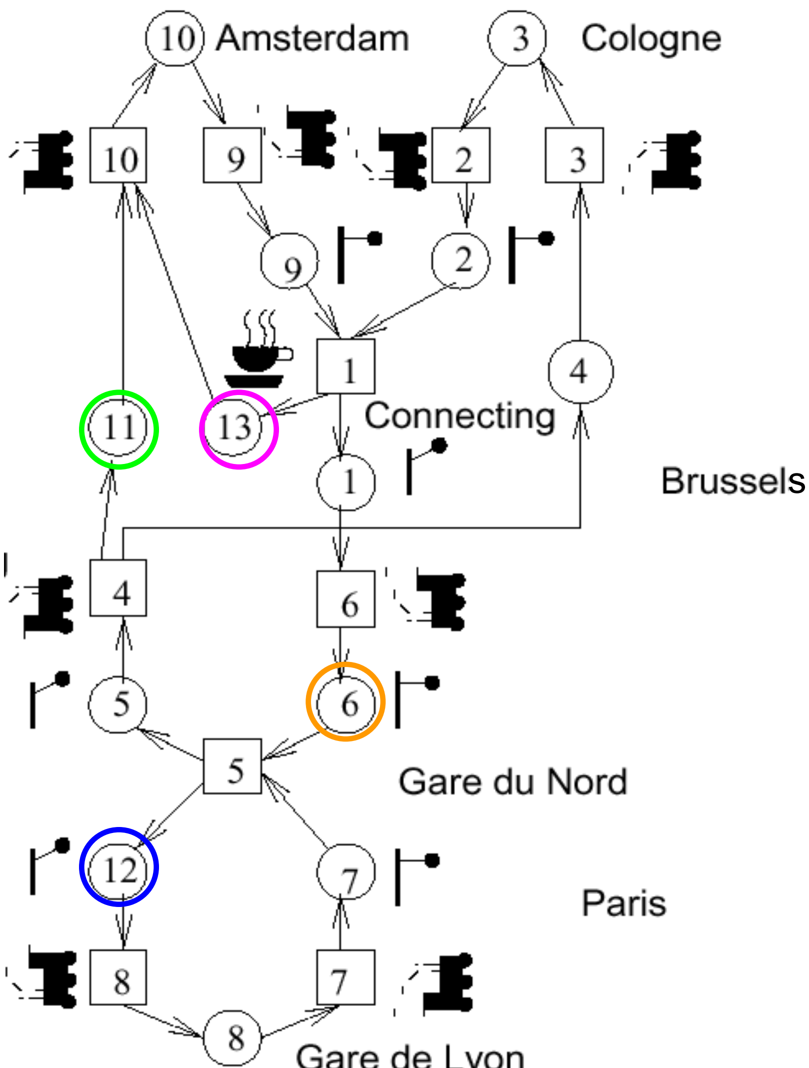
Application to Thalys example

$\underline{N}^T\, c_R = 0$,
with $\underline{N}^T$ =

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}
t_1	1	-1							-1				1
t_2		1	-1										
t_3			1	-1									
t_4				1	-1						1		
t_5					1	-1	-1					1	
t_6	-1					1							
t_7							1	-1					
t_8								1				-1	
t_9									1	-1			
t_{10}										1	-1		-1

- ∑rows=0 → 1 linear dependency among rows
- → rank = 10-1 = 9
  - Dimension of solution space = 13 - rank = 4
- Solutions? Educated guessing

Manually Finding Basis Vectors



- Set 1 of 4 components = 1, others = 0
  - Find independent sets components in the matrix after deleting one row
    - *E.g.*, consider places 7, 8, 12

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}
t_1	1	-1							-1				1
t_2		1	-1										
t_3			1	-1									
t_4				1	-1						1		
t_5					1	-1	-1					1	
t_6	-1					1							
t_7							1	-1					
t_8								1				-1	
t_9									1	-1			
t_{10}										1	-1		-1

  - Or assume that a particular place is included in R
    - E.*g.* p_6
  - whereas places along the path of other objects are not
    - E.*g.* p_{11}, p_{12}, p_{13}

6=delete 1st row

1st Basis Vector

Set one component (6, 11, 12, 13) to 1, others to 0.
$\rightarrow$ **1st basis** b_1:
$b_1(p_6)$=1, $b_1(p_{11})$=0,
$b_1(p_{12})$=0, $b_1(p_{13})$=0

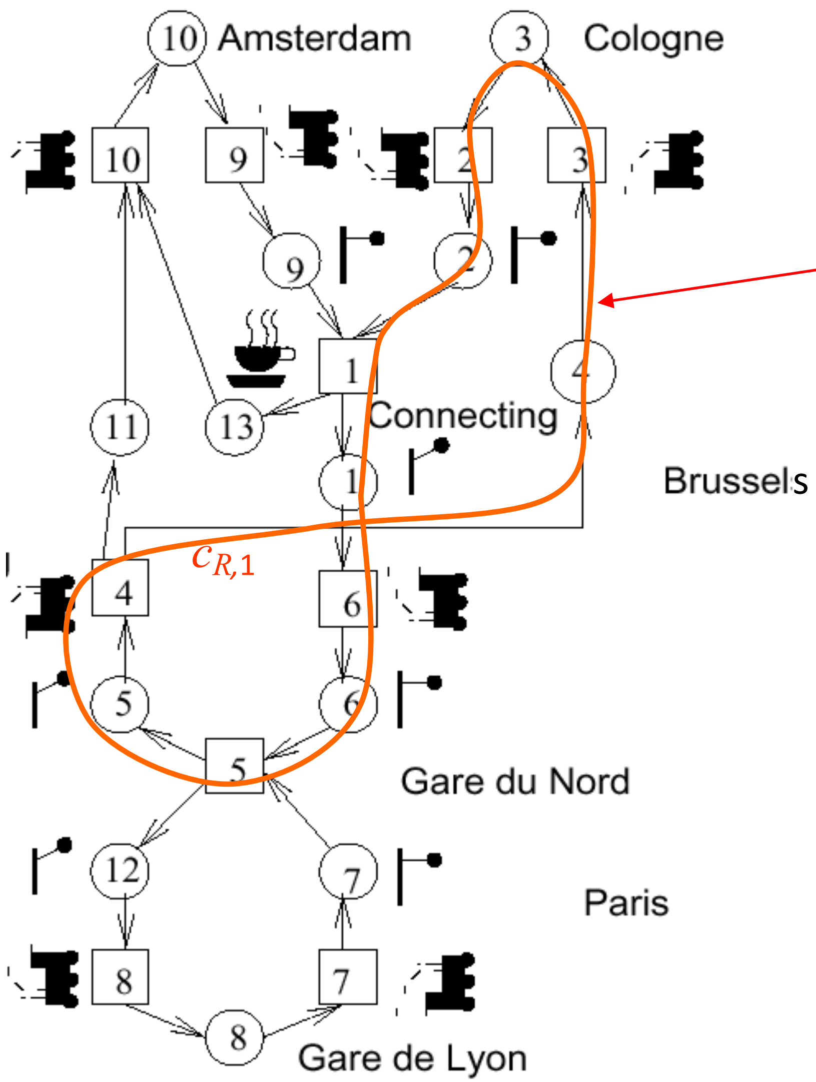
	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}
t_1	1	-1							-1				1
t_2		1	-1										
t_3			1	-1									
t_4				1	-1						1		
t_5					1	-1	-1					1	
t_6	-1					1							
t_7							1	-1					
t_8								1				-1	
t_9									1	-1			
t_{10}										1	-1		-1

- $t_{10}(p_{10})\ b_1(p_{10}) + t_{10}(p_{11})\ b_1(p_{11}) + t_{10}(p_{13})\ b_1(p_{13}) = 0$
  $\rightarrow b_1(p_{10}) = 0$
- $t_9(p_9)\ b_1(p_9) + t_9(p_{10})\ b_1(p_{10}) = 0$
  $\rightarrow b_1(p_9)=0$
- ...

b_1 = (1,1,1,1,1,1,0,0,0,0,0,0,0)

All components $\in$ {0, 1}
$\rightarrow$ $\boldsymbol{c_{R1}} = b_1$

Interpretation of the 1st Invariant



$$c_{R,1} = \left(1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\right)$$

- Characteristic vector describes places for the Cologne train

We proved that:

- The number of trains along the path remains constant ☺

2nd Basis Vector

Set one component (6, 11, 12, 13) to 1, others to 0.

$\rightarrow$ **2nd basis** b_2:

$\rightarrow b_2(p_6)=0$, $b_2(p_{11})=1$, $b_2(p_{12})=0$, $b_2(p_{13})=0$

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}
t_1	1	-1							-1				1
t_2		1	-1										
t_3			1	-1									
t_4				1	-1						1		
t_5					1	-1	-1					1	
t_6	-1					1							
t_7							1	-1					
t_8								1				-1	
t_9									1	-1			
t_{10}										1	-1		-1

- $t_{10}(p_{10})\, b_2(p_{10}) + t_{10}(p_{11})\, b_2(p_{11}) + t_{10}(p_{13})\, b_2(p_{13}) = 0$
  $\rightarrow b_2(p_{10}) = 1$
- $t_9(p_9)\, b_2(p_9) + t_9(p_{10})\, b_2(p_{10}) = 0$
  $\rightarrow b_2(p_9)=1$
- ...

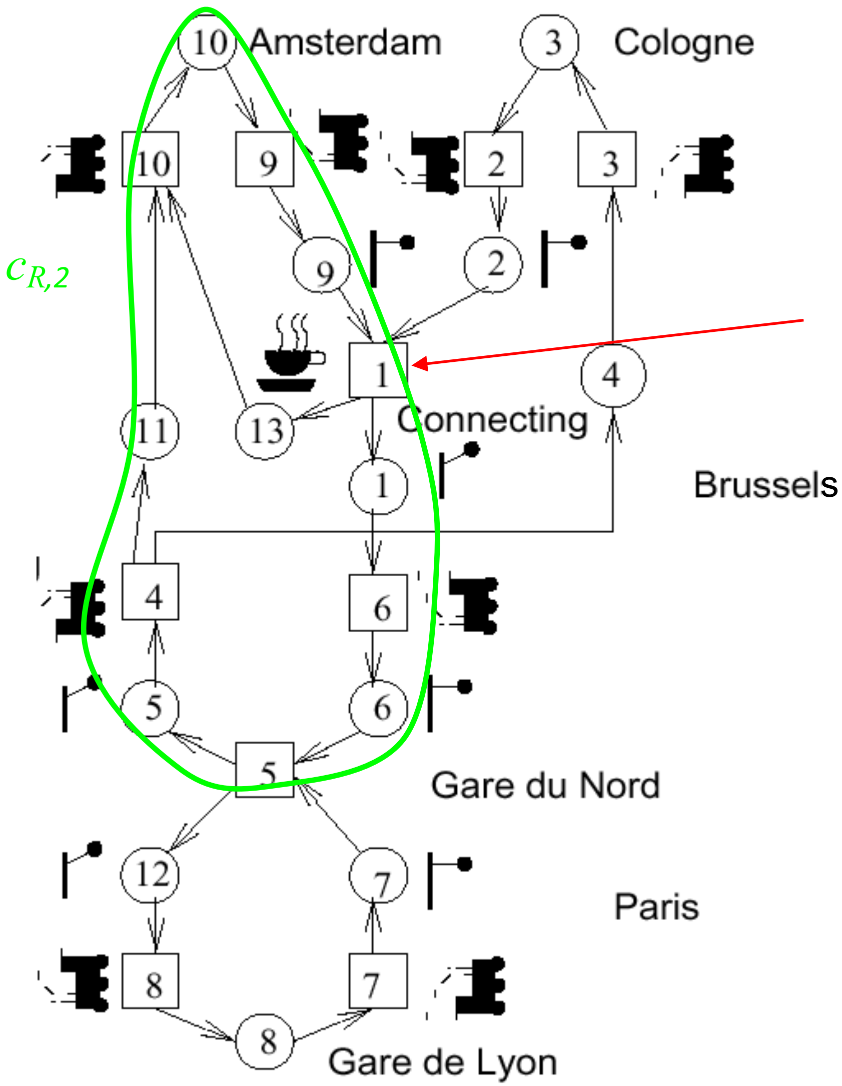
$b_2 = (0,-1,-1,-1,0,0,0,0,1,1,1,0,0)$

$b_1 = (1, 1, 1, 1,1,1,0,0,0,0,0,0,0)$

b_2 not a characteristic vector, but $\boldsymbol{c_{R,2}}=b_1+b_2$ is

$\rightarrow \boldsymbol{c_{R,2}} =$ (1,0,0,0,1,1,0,0,1,1,1,0,0)

Interpretation of the 2nd Invariant

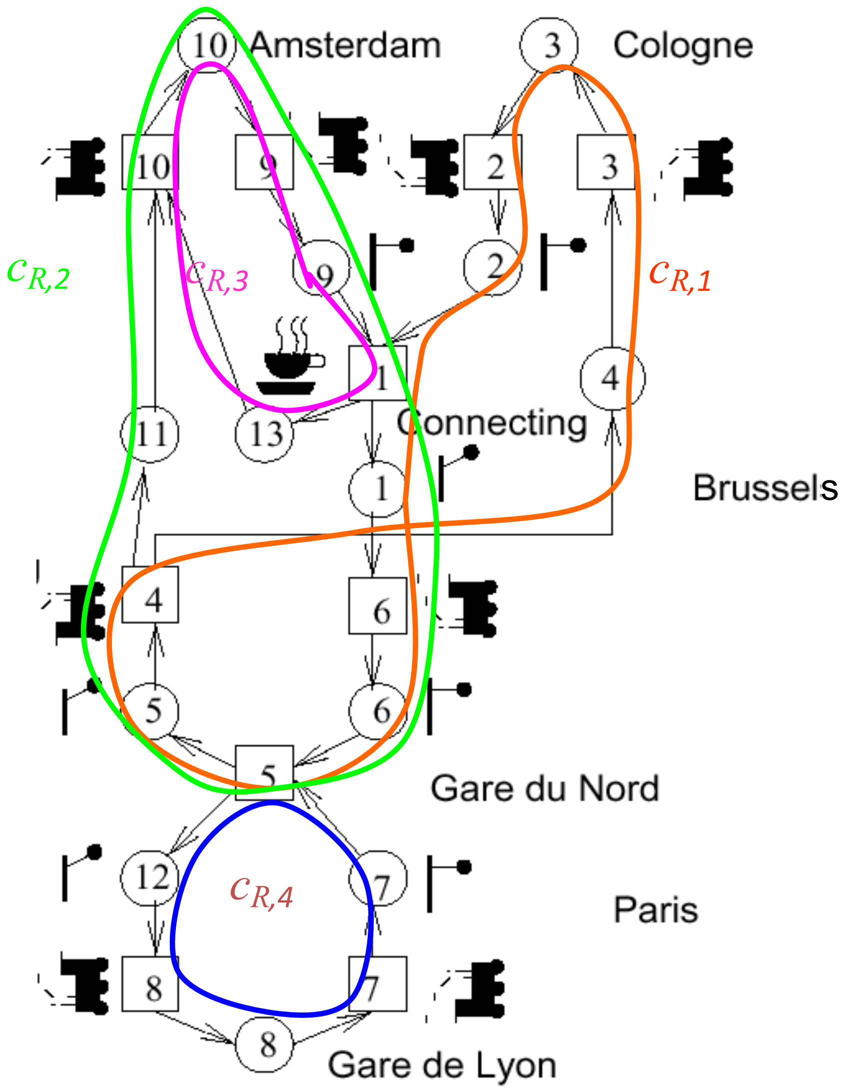


$c_{R,2} = (1,0,0,0,1\,,1,0,0,1,1\,,1,0,0)$

We proved that:

- None of the Amsterdam trains get lost (also nice to know ☺)

Two More Invariants Later



$$c_{R,1} = (1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$$

$$c_{R,2} = (1\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0)$$

$$c_{R,3} = (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1)$$

$$c_{R,4} = (0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 0)$$

We proved that:

- The number of trains serving Amsterdam, Cologne and Paris remains constant, and
- the number of train drivers remains constant

Applications

- Modeling of (shared) resources
- Modeling of mutual exclusion
- Modeling of synchronization

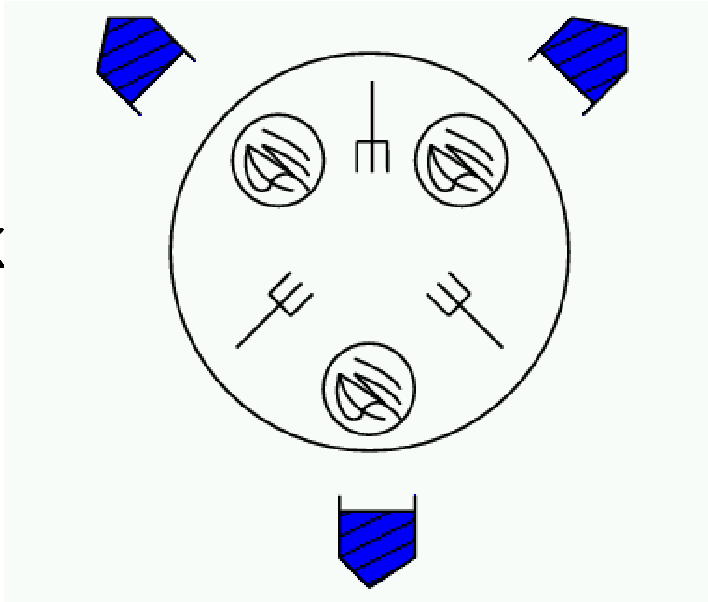
Predicate/Transition Nets

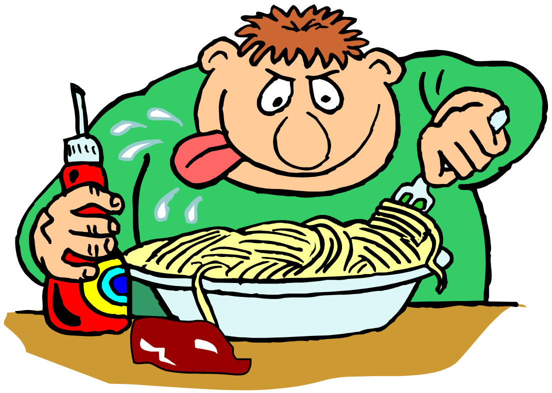
- *Goal*: compact representation of complex systems
- Key changes
  - Tokens become *individuals*
  - Transitions are enabled if *functions* on incoming edges are satisfied
  - The number of individuals generated by firing transitions is defined with functions
- Changes can be explained by un/folding C/E nets

  $\Rightarrow$ Semantics can be defined by C/E nets

Example: Dining Philosophers

- $n > 1$ philosophers sitting at a round table;
- n forks,
- n plates with spaghetti;
- philosophers either think or eat spaghetti (using left and right fork)



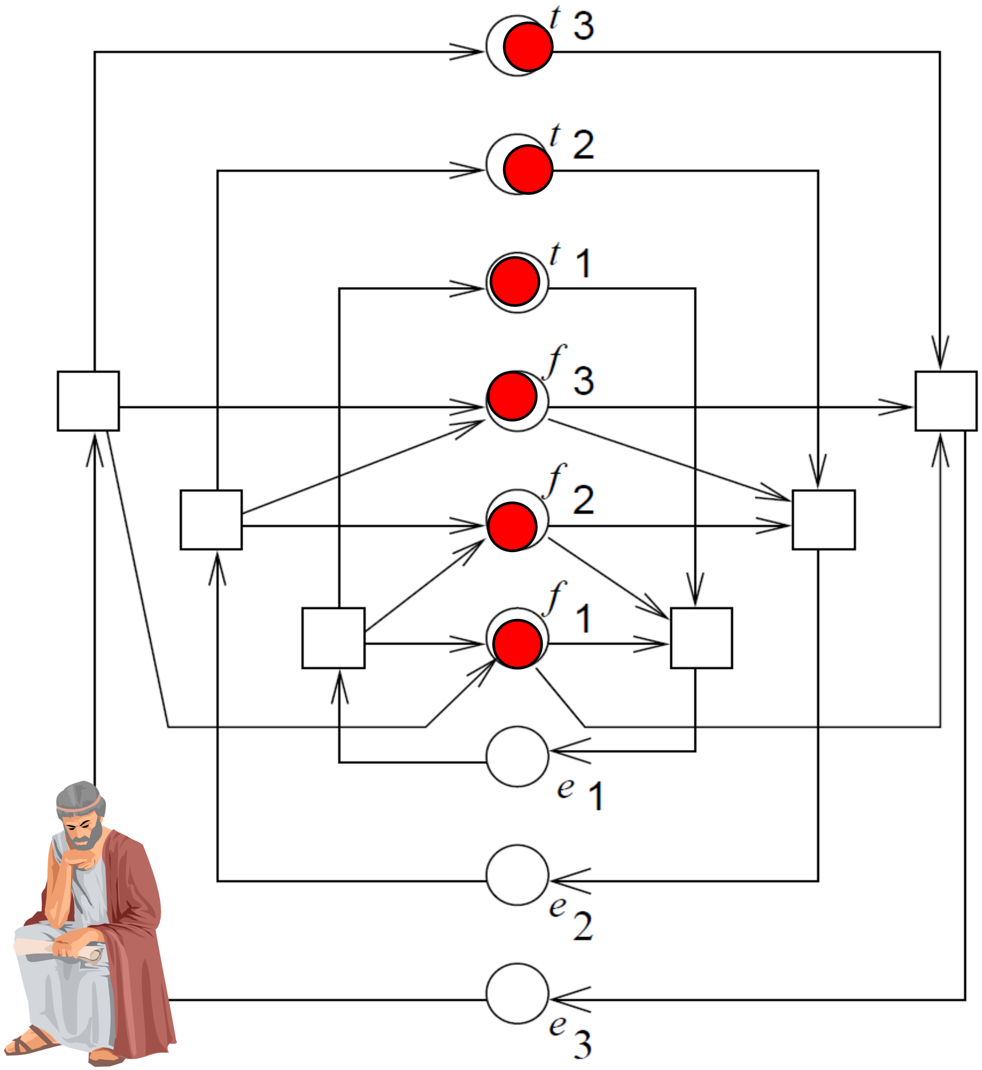


2 forks needed!

How to model conflict for forks?

How to guarantee there is no starvation?

C/E Model of the Dining Philosophers



Let $x \in \{1..3\}$

t_x: x is thinking

e_x: x is eating

f_x: fork x is available

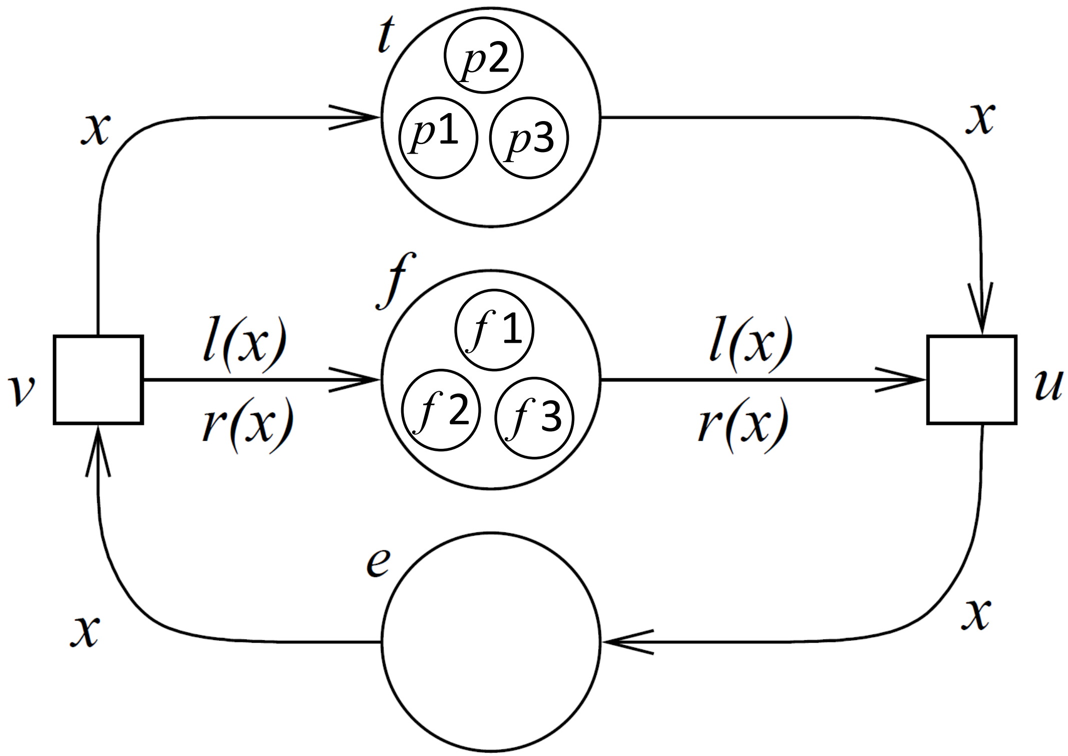
The model is quite clumsy, and it is difficult to extend to more philosophers!

Alternative Predicate/Transition Model

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.

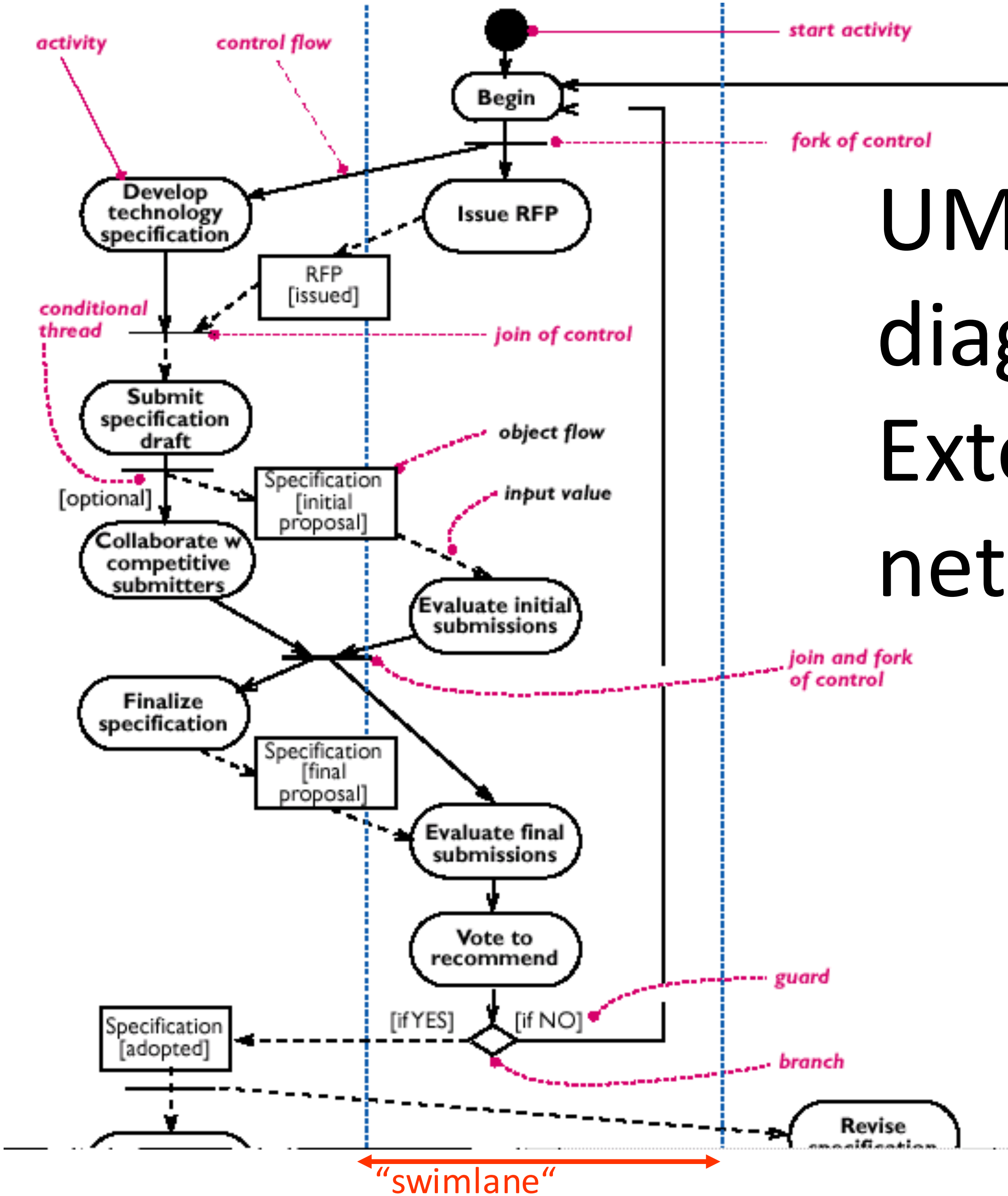
Semantics can be defined by replacing the net by the equivalent condition/event net.

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 effort to remove limitations

UML Activity diagrams: Extended Petri nets

- Include decisions (like in flow charts)
- Graphical notation similar to SDL



© Cris Kobryn: UML 2001: A Standardization Odyssey, CACM, October, 1999

Summary

- Petri nets: focus on causal dependencies
  - 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
- Activity diagrams in UML are extended Petri nets

Next Time

- Discrete Event Modeling
  - VHDL
  - Multi-value logic
  - Chapter 2.7