PhD Summer School
Formal Methods for System Analysis in Informatics
Druskininkai, LT, May 2007

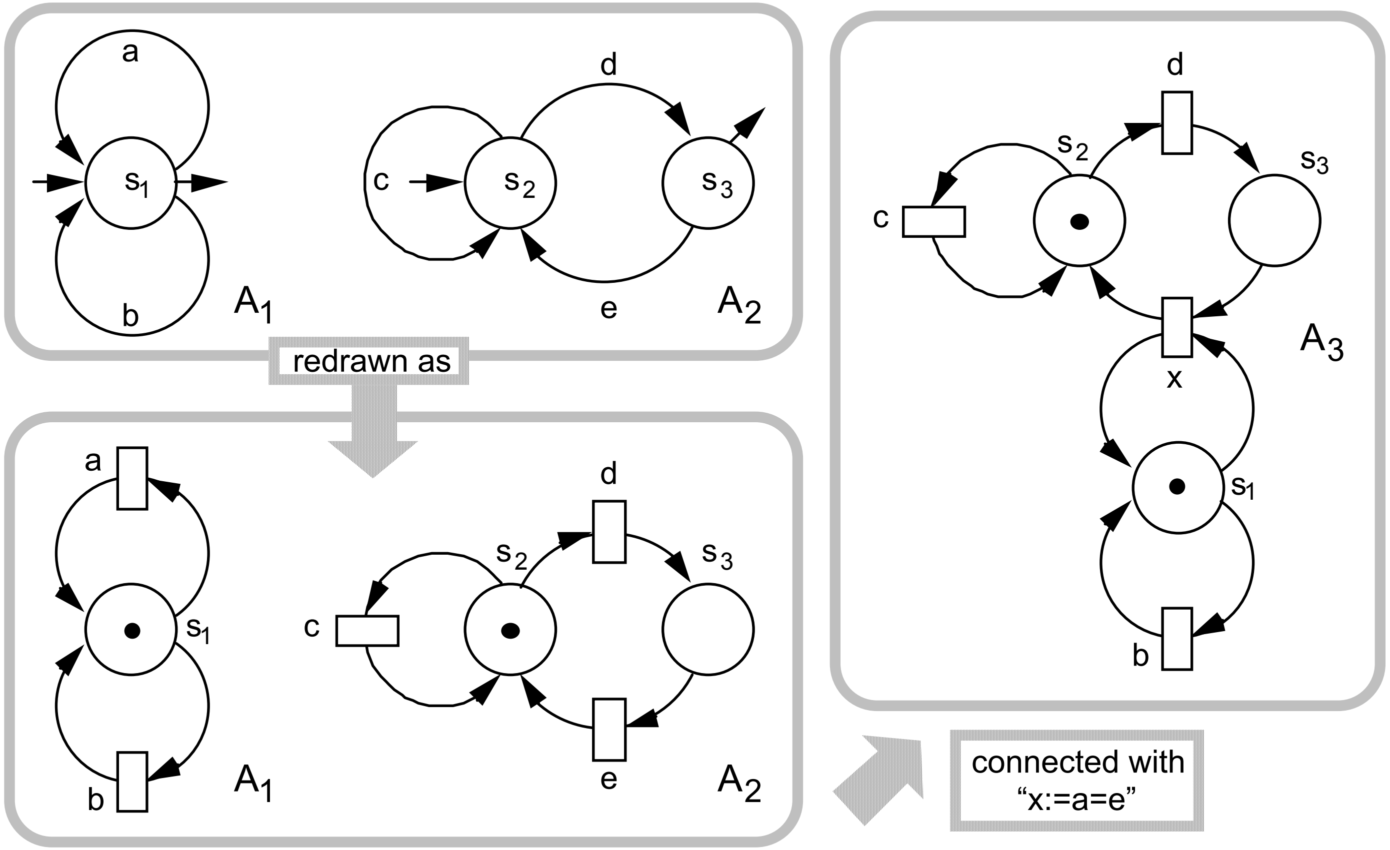
Petri Net basics

Bernd Baumgarten

Fraunhofer SIT
Rheinstr. 75, 64295 Darmstadt, Germany
bernd.baumgarten@sit.fraunhofer.de
http://private.sit.fhg.de/~baumgart/
+49 6151 869263

Automata performing common transitions

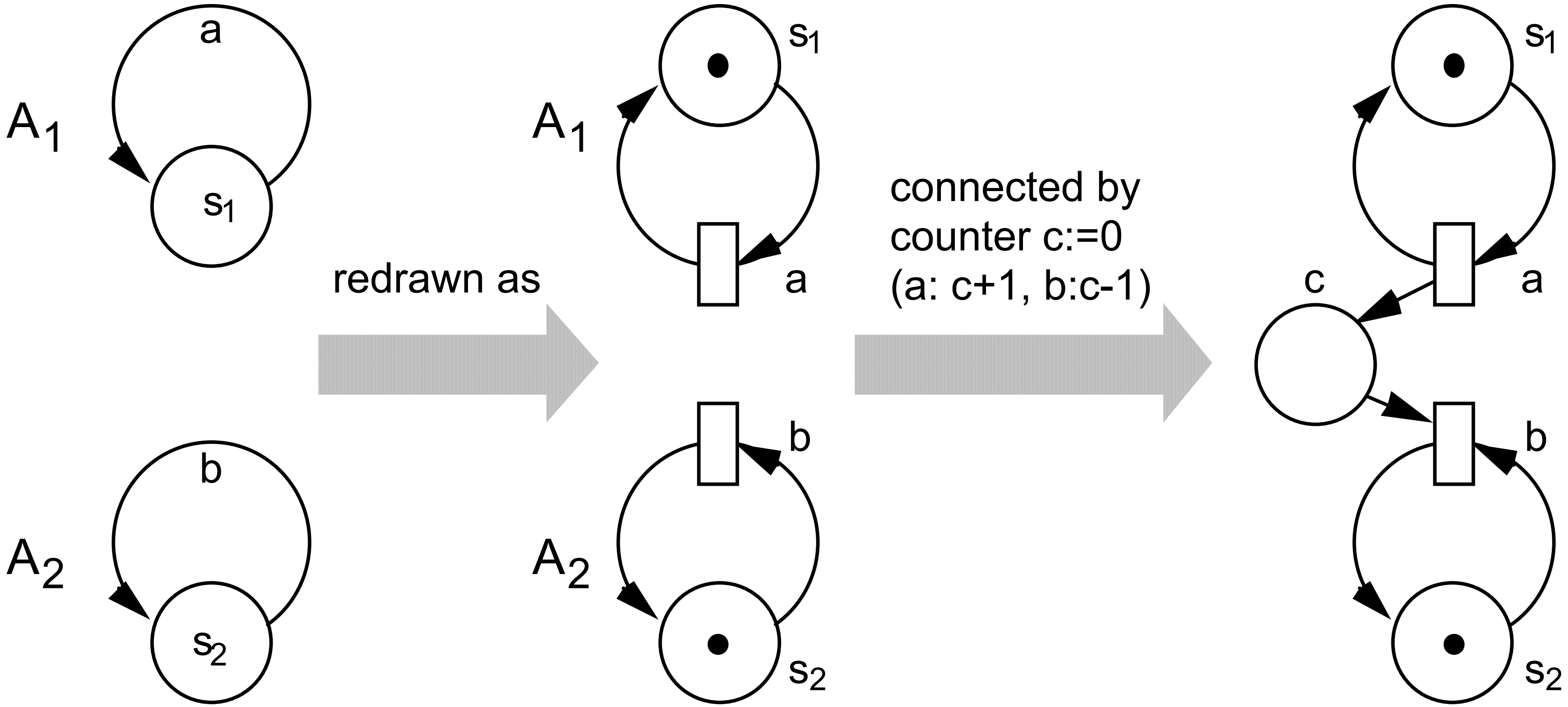
We can "glue the automata together" at transitions.



Two automata coupled by a common transition

Automata coupled by counters

We may connect automata via counting up and down “shared counters.”



Producer and consumer

Advantages

Transition-coupled finite automata

are a short and intuitive way

to represent larger *finite* automata in a compact manner.

Finite automata coupled via counters

are a short and intuitive way

to represent larger *finite or infinite* automata in a compact manner.

common generalization → **Petri Nets**
(Place-Transition Systems)

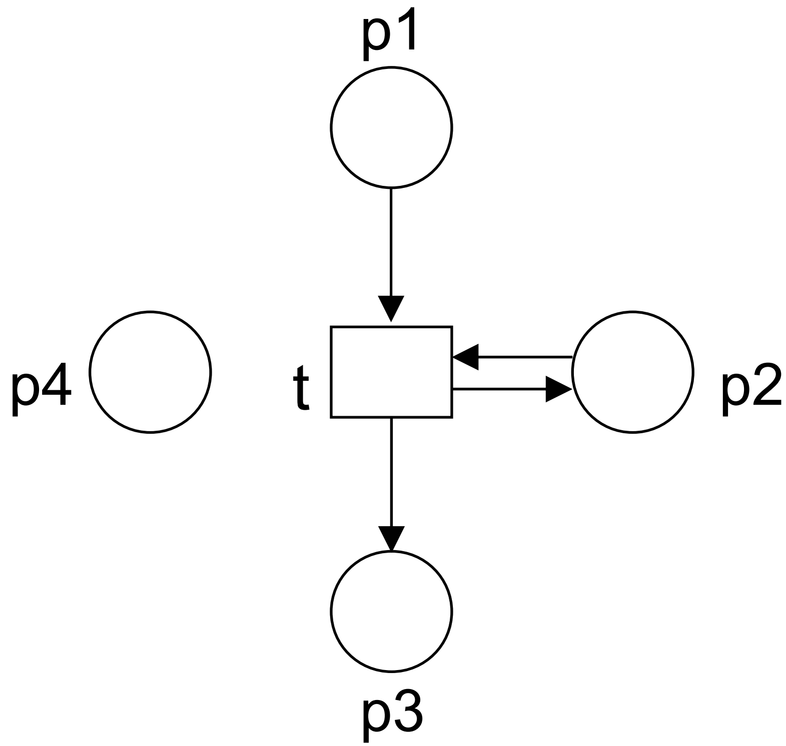
Net graphs

A **net** or **net graph**

is a triple $N = (P,T,F)$ such that
$P \cap T = \varnothing$ and
$F \subseteq (P \times T) \cup (T \times P)$.

Elements of P: **places**, represented by circles;
elements of T: **transitions**, repr'ed by bars or rectangles;
} **nodes**

elements of F: **arcs**, represented by arrows.



net ({p1, p2, p3, p4},
{ t },
{(p1;t), (p2,t), (t,p2), (t,p3)})

Notions in net graphs

Pre-set of node (place or transition) x:

$\bullet x := \{ y \mid (y,x) \in F \}$,

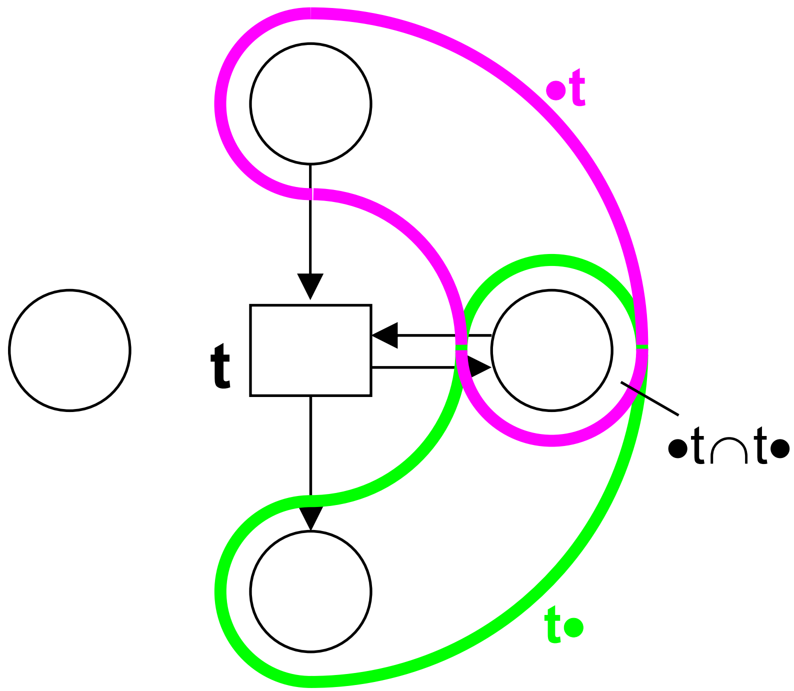
the set of all **input nodes**
(transitions, places, respectively).

Post-set of node (place or transition) x:

$x \bullet := \{ y \mid (x,y) \in F \}$,

the set of all **output nodes**
(transitions, places, respectively).

A **loop** of N is a subset $\{(s,t),(t,s)\} \subseteq F$.



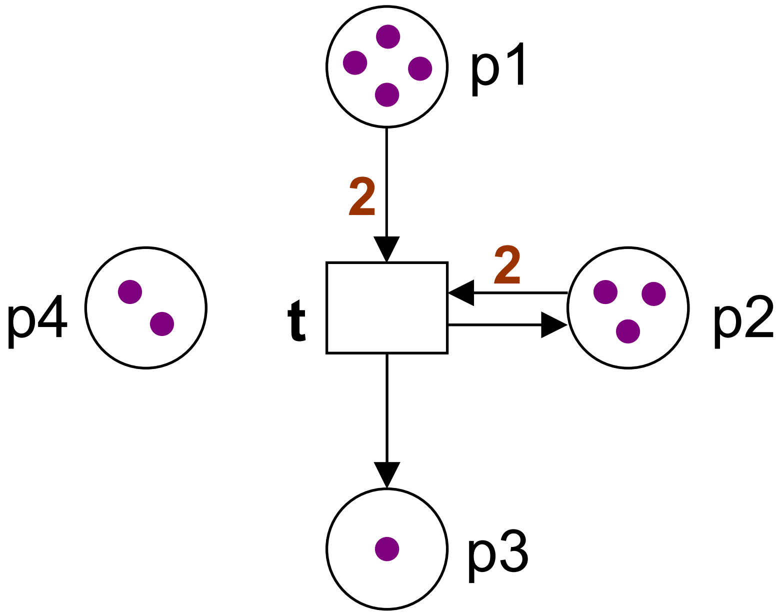
PT systems

A 5-tuple $S = (P,T,F,W,M_0)$ is called a **place-transition system** or **PT system**, if

(P,T,F) is a net,
$W : F \to \mathbb{N}$ (**arc weights**), and
$M_0 : P \to \mathbb{N}_0$ (**initial marking**, consisting of "**tokens** on places").

default: 1

dots



Markings, Activation, Firing

Marking : $M: P \to \mathbb{N}_0$

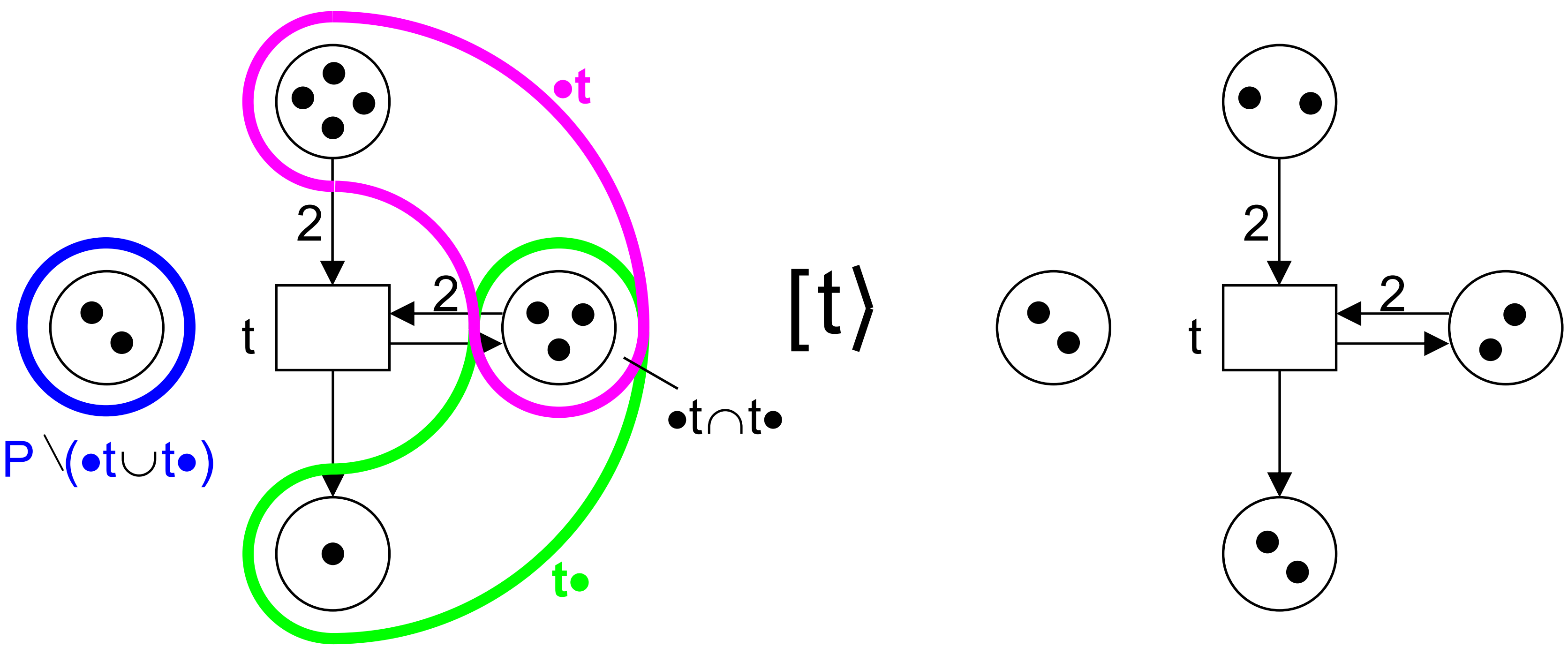
t is **activated** or **enabled** under M, written as $M[t\rangle$:

$$\forall p \in {}^\bullet t: M(p) \geq W(p,t).$$

If $M[t\rangle$, then t can **occur** (or **fire**),
changing M into the **follower marking** Mt,
written as $M\,[t\rangle\, Mt$:

$$Mt(p) := \begin{cases} M(p) - W(p,t) & \text{if } p \in {}^\bullet t \setminus t^\bullet \\ M(p) + W(t,p) & \text{if } p \in t^\bullet \setminus {}^\bullet t \\ M(p) - W(p,t) + W(t,p) & \text{if } p \in {}^\bullet t \cap t^\bullet \\ M(p) & \text{else}. \end{cases}$$

Transition occurrence – an Example



The long range dynamics of PT systems

Let $S = (P,T,F,W,M_0)$ be a PT system. We call a marking M **reachable from** M_0 **by** a transition sequence $w = t_1 \ldots t_n \in T^*$ and write $M_0[w\rangle M$ if

either $w = \varepsilon \wedge M = M_0$

or $\exists$ marking $M' : M_0[t_1 \ldots t_{n-1}\rangle M' \wedge M'[t_n\rangle M$.

In this case, w is called a **firing** (or **occurrence**) **sequence**,

we call w **activated** under M_0 and write $M_0[w\rangle$.

The marking M reached after w is denoted by $M_0 w$.

$Occ(S) := \{w \in T^* \mid M_0[w\rangle\}$: the set of all firing sequences.

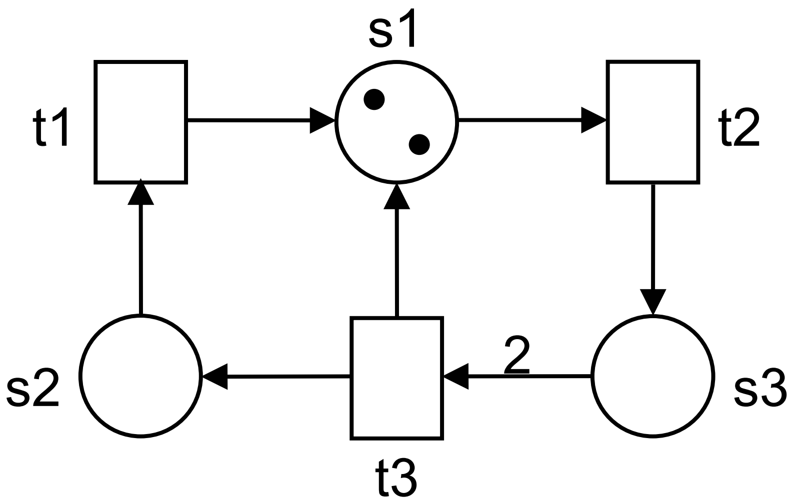
$Reach(S) := \{ M_0 w \mid w \in Occ(S)\}$: the **reachability set** of S.

If v and w are firing sequences and permutations of each other, then

$$M_0 w = M_0 v.$$

Net Dynamics – Example

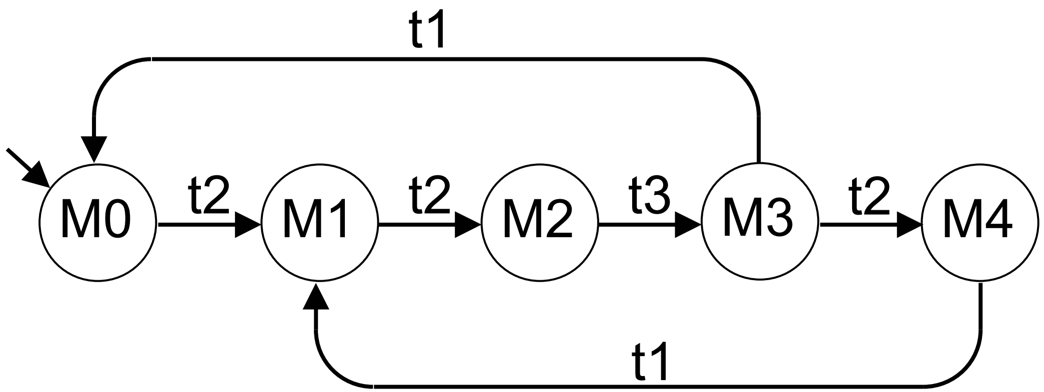
PT-System *Sys*



Analysis ➔ *Reach*(*Sys*)

	s1	s2	s3	Occurrences
M0	2	0	0	t2→M1 ✓
M1	1	0	1	t2→M2 ✓
M2	0	0	2	t3→M3 ✓
M3	1	1	0	t1→M0, t2→M4 ✓
M4	0	1	1	t1→M1 ✓

Analysis ➔ *Occ*(*Sys*)

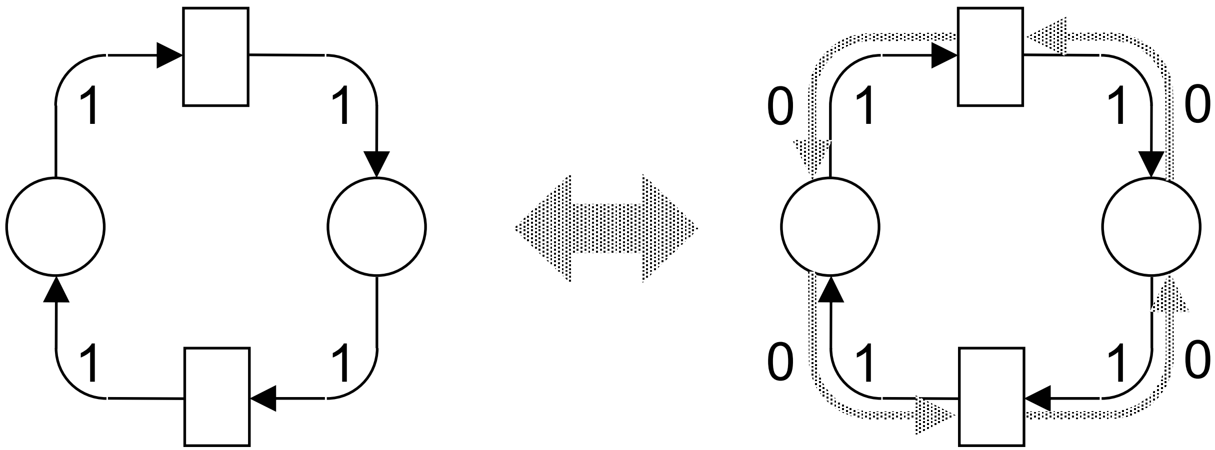


Alternative definitions of PT systems

Expressing *F* by *W*

PT system given by 4-tuple $S = (P,T,W,M_0)$, i.e. without the explicit set *F* of arcs:

> $W : (P \times T) \cup (T \times P) \to \mathbb{N}_0$, and arcs with $W(x,y) = 0$ are simply not drawn!



Now, the **transition firing** effect is **elegantly** expressed by a **single case**:

$$M(p) - W(p,t) + W(t,p).$$

Net languages

(**Transition**) **labelled** PT system (S,h):

- PT system$S = (P,T,F,W,M_0)$
- labelling $h: T \to A \cup \{\varepsilon\}$

(S,h) defines a **label language**: “write down the labels of firing transitions”

$$L(S,h) := H(Occ(S)) \text{ , where } H : \begin{cases} Occ(S) & \to & A^* \\ \varepsilon & \mapsto & \varepsilon \\ wt & \mapsto & H(w)h(t) \end{cases} .$$

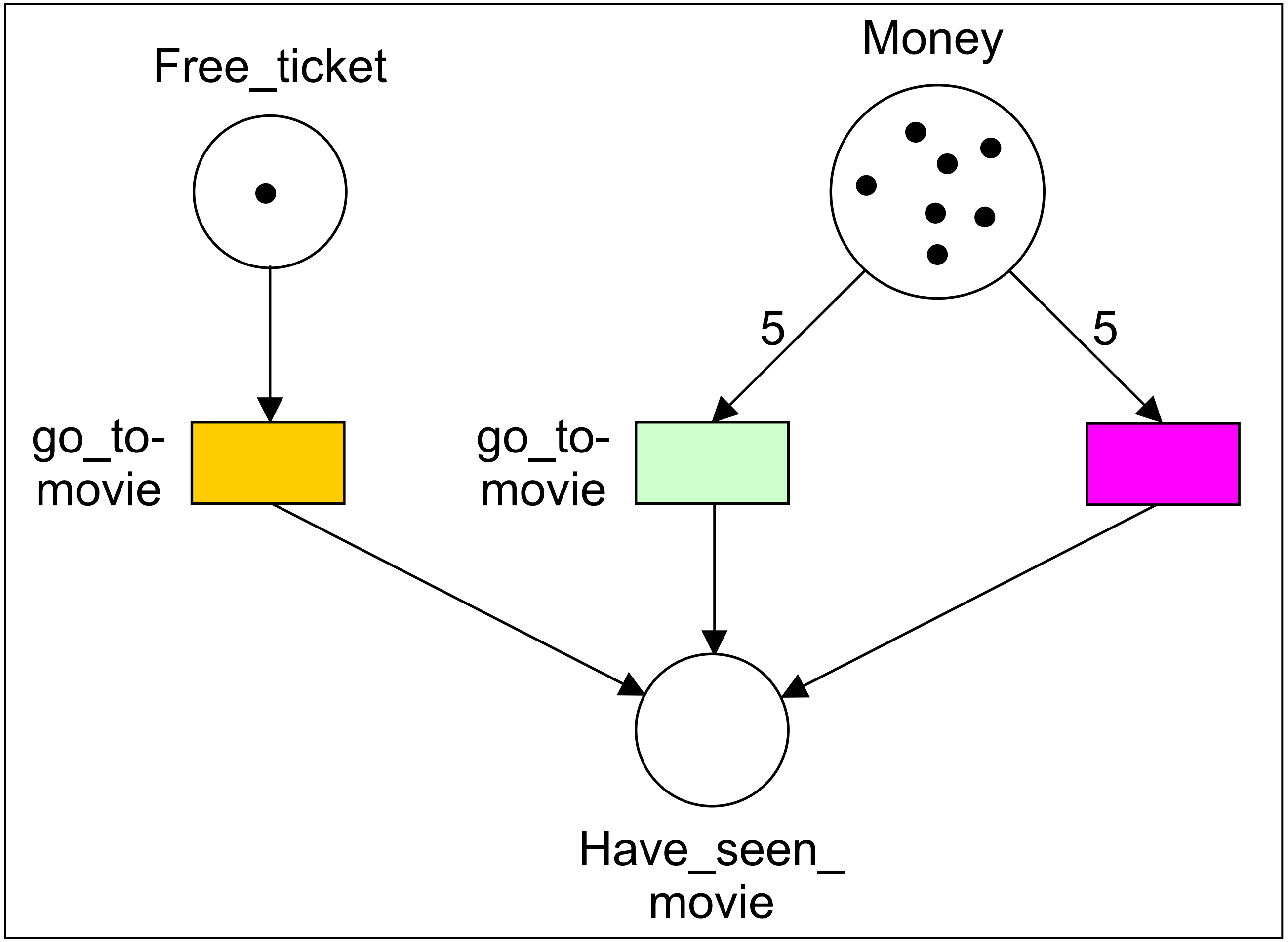
Label languages are prefix-closed.

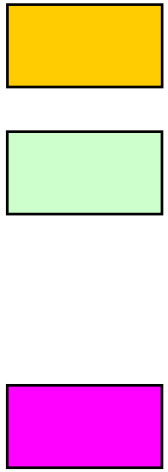
h = identity → $Occ(S)$.

Exercise:

Find a labelled PT system with the label language $\{a^n b^m \mid 0 \le m \le n\}$.

A labelled net: I go to the cinema





cannot be distinguished by people seeing me sitting in the cinema

cannot be observed, as I go in disguise/ incognito.

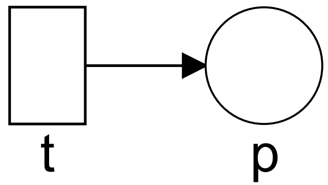
Analysis of Place-Transition Nets (1)

All nets considered in this chapter are finite.

1. Determine *Reach*(*S*).

Reach(*S*) finite $\Rightarrow$ BFRA algorithm produces a tabular listing.

Reach(*S*) infinite $\Rightarrow$? Example of a simple case:



$M \in$ *Reach*(*S*) $\Rightarrow$ BFRA algorithm produces M — sooner or later!

2. Find out whether *Reach*(*S*) is finite or not.

BFRA is only a semi-decision-procedure; it works only if YES.
However, the coverability analysis (CA) algorithm will <u>always</u> tell us.

Analysis of Place-Transition Nets (2)

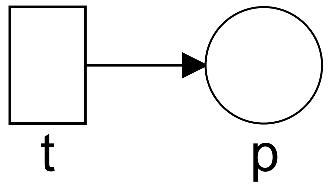
3. For a given marking *M*, find out if *M* ∈ *Reach*(*S*)

This is the **reachability problem**.
In the infinite case BFRA is only a semi-decision-procedure.
A decision algorithm was found, but it is very complex.

4. Determine *Occ*(*S*).

Reach(*S*) finite ⇒ RG(*S*) is a finite acceptor for *Occ*(*S*).

Reach(*S*) infinite ⇒ ? Example of a simple case:



Partial information about *Occ*(*S*) from CA: $\{t \in T \mid \forall w \in Occ(S) : \#(t,w) = 0\}$, the set of **dead transitions.**

Analysis of Place-Transition Nets (3)

5. The computation of P-invariants

… is a part of the “linear analysis” of PT systems

It yields properties of all reachable markings, even in the case of infinite *Reach*(*S*).

Moreover it does so for arbitrary initial markings,
and thus for infinitely many PT systems at the same time.

Reachability analysis and boundedness

S is called **bounded** if $\exists b \in I\!N : \forall M \in Reach(S), p \in P : M(p) \leq b$.

A PT system *S* is **bounded** if and only if *Reach*(*S*) is **finite**.

The **reachability graph** *RG*(*S*) of *S* is the rooted (usually neither minimal nor complete) acceptor for *Occ*(*S*) with

- alphabet *T*,
- state set *Reach*(*S*),
- transition function $\delta(M,t) := Mt$ (if $M[t\rangle$),
- initial state M_0, and
- terminal state set *Reach*(*S*).

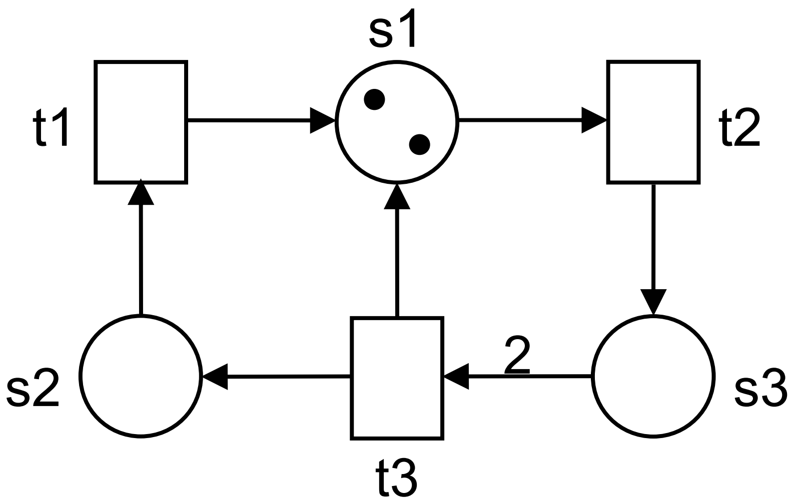
It can computed in tabular form by **breadth-first reachability analysis** (**BRFA**).

A PT system *S* is bounded if and only if **BFRA terminates**.

Every reachable marking (state) and transition occurrence (state transition) in the reachability graph is eventually produced by BFRA, if performed long enough.

We already saw this BFRA example

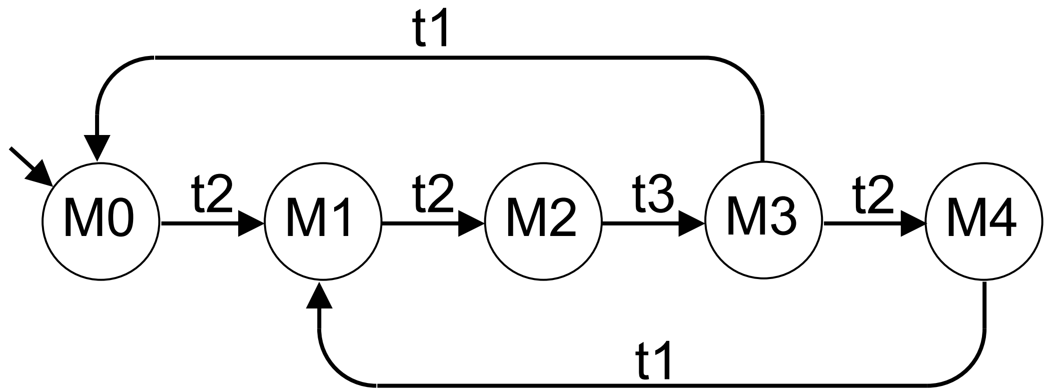
PT-System *Sys*



Analysis ➔ *Reach*(*Sys*)

	s1	s2	s3	Occurrences
M0	2	0	0	t2→M1 ✓
M1	1	0	1	t2→M2 ✓
M2	0	0	2	t3→M3 ✓
M3	1	1	0	t1→M0, t2→M4 ✓
M4	0	1	1	t1→M1 ✓

Analysis ➔
RG(Sys), *Occ*(*Sys*)



Coverability analysis

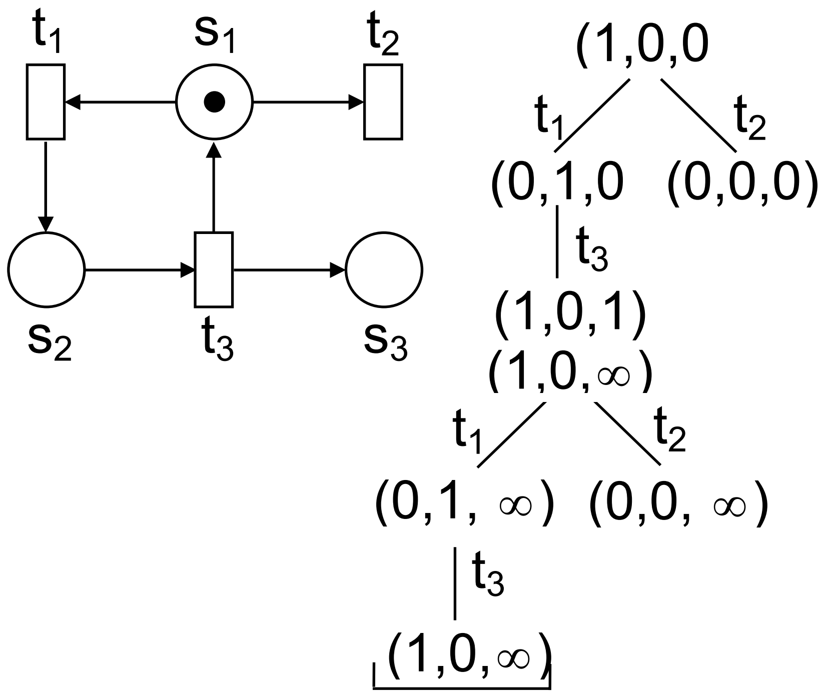
A **quasi-marking** is a “marking that may take the value infinity”, i.e. a mapping $Q{:}P \rightarrow I\!N_0 \cup \{\infty\}$.

The **coverability tree** *CovTr*(*S*) is the labelled tree defined by the **coverability analysis** algorithm CA.

We represent a labelled tree by the set of its ordered pairs
(node (=state), word accepted on the path).

The **coverability tree** *CovTr*(*S*) is the labelled tree defined by the **coverability analysis** algorithm CA.

Example



- similar to RA, but a tree
- stop at repetitions
- raise quasi-marking to ∞ if $\geq$ a previous one, wherever strictly grown
- calculate with ∞ “as usual”

Coverability analysis, properties

CA always **terminates**. *CovTr*(*S*) is always **finite**.

S is **bounded** if and only if
CA never produces the quasi-marking value ∞.

A transition *t* is **dead** in *S* if and only if
it does not label any arc of the coverability tree.

Even more facts can be obtained from *CovTr*(*S*), cf. literature.

Linear analysis: Matrix representation of a net

We assume that S = (P,T,F,W,M_0) is a PT system and
that in the PT net N = (P,T,F,W,)
$P = \{p_1, p_2, \ldots, p_m\}$ and $T = \{t_1, t_2, \ldots, t_n\}$.

The $m \times n$ **incidence matrix** $C = Inc(N)$ of S is defined by

$$\forall 1 \le i \le m, 1 \le j \le n : C_{ij} := W(t_j, p_i) - W(p_i, t_j).$$

(Consider W as defined on $(P \times T) \cup (T \times P)$, letting $W(x,y):=0$ for $(x,y) \notin F$.)

If the PT net N is **loop-free**, then it is **uniquely determined** by $Inc(N)$.

Linear analysis: Basic equation

As C_{ij} tells how the token number on place p_i will change if transition t_j occurs,

$C_{\bullet j}$, the *j*-th column of *C*,

shows the change of the entire marking if transition t_j occurs:

$$Mt_j = M + C_{\bullet j}.$$

Now we associate with every transition sequence *w* the number of the occurrences of each transition in *w* and list these numbers in the **Parikh vector** $\overline{w}$ of *w:*

$$\overline{w} := \begin{pmatrix} \#(t_1, w) \\ \vdots \\ \#(t_2, w) \end{pmatrix}$$

⇒

Basic equation of linear analysis

If *w* is an occurrence sequence of *S* and *C*=*Inc*(*N*), then

$$M_0 w = M_0 + C\overline{w}.$$

Linear analysis, P-invariants

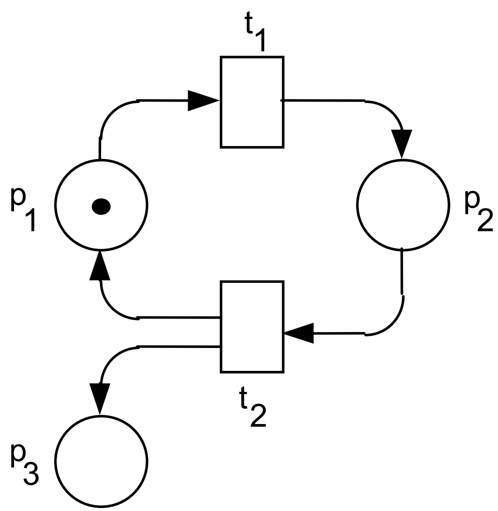
P-invariant: an $x \in Int^m$ with $C^{\mathsf{T}} x = 0$

> *Constance of markings weighted with P-invariant*
>
> An *m*-tuple $x \in Int^m$ is a P-invariant of a PT net *N* if and only if
>
> for all possible initial markings *M*:
>
> $$\forall M' \in Reach(N,M): M' \cdot x = M \cdot x.$$

Why? "$\Rightarrow$" is simple:

$$\begin{aligned} M' \cdot x &= M'^{\mathsf{T}} x \\ &= (M + C\overline{w})^{\mathsf{T}} x \\ &= M^{\mathsf{T}} x + (C\overline{w})^{\mathsf{T}} x \\ &= M \cdot x + \overline{w}^{\mathsf{T}} (C^{\mathsf{T}} x) \\ &= M \cdot x \end{aligned}$$

Linear analysis, example



This PT system has the following incidence matrix:

$$\begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{cc} t_1 & t_2 \end{array} \\ \begin{pmatrix} -1 & 1 \\ 1 & -1 \\ 0 & 1 \end{pmatrix} \end{array}.$$

The system of linear equations

$$\begin{pmatrix} -1 & 1 \\ 1 & -1 \\ 0 & 1 \end{pmatrix}^T \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -1 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \text{ i.e. } \begin{array}{l} -x_1 + x_2 = 0 \\ x_1 - x_2 + x_3 = 0 \end{array}, \text{ yields e.g. } \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

as one P- invariant, hence (no surprise):

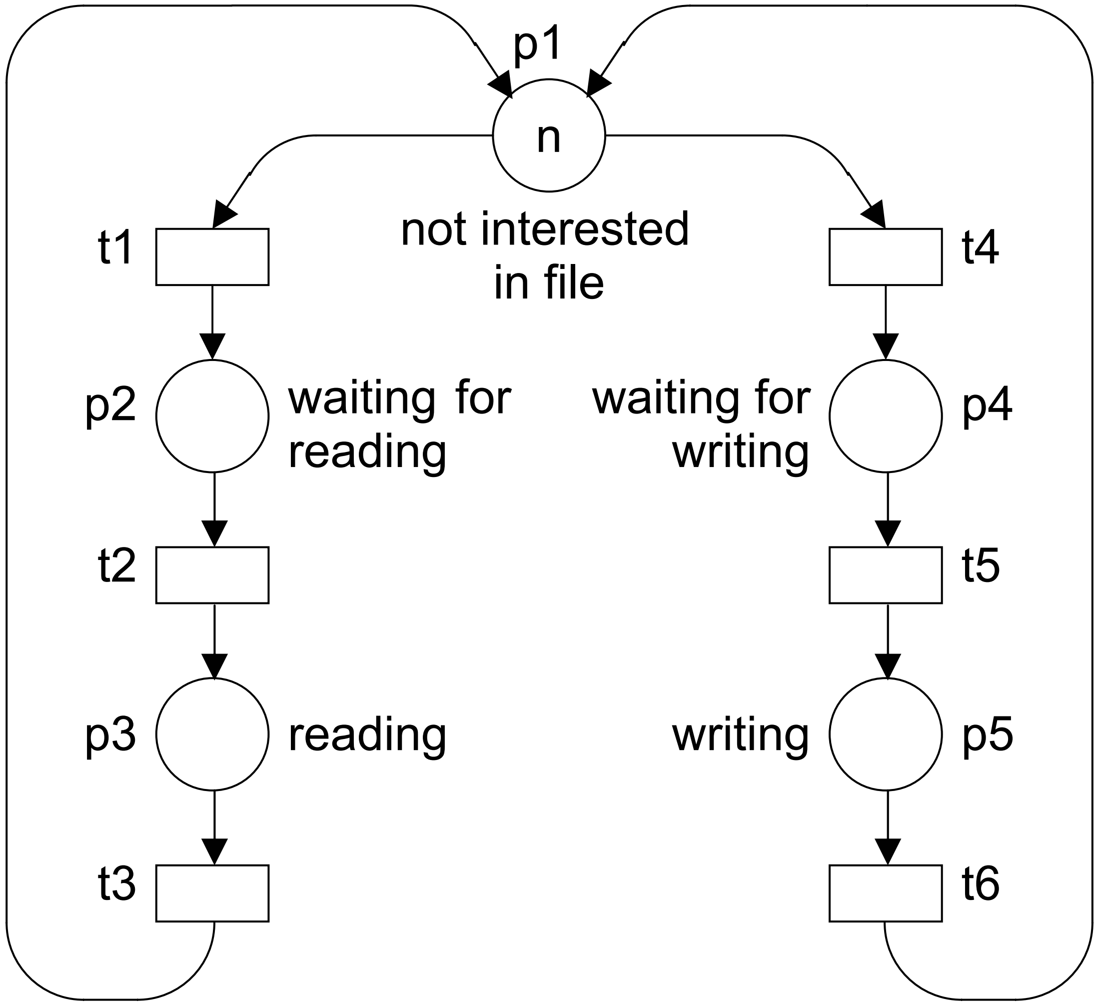
$$\forall M \in Reach(S_u) : M(p_1) + M(p_2) = M_0(p_1) + M_0(p_2) = 1.$$

A nice application of linear analysis: readers-writers problem

- *n* processes may access a file for reading or writing.
- They want to coordinate their accesses such that
  - **several processes may read** at the same time;
  - while **one process writes**, **no-one else** may have **access**.

→ Problem as PT-system shown in **black**
(parameterized scheme, for any *n*)

Problem net


p1
n
not interested in file
t1
t4
p2
waiting for reading
waiting for writing
p4
t2
t5
p3
reading
writing
p5
t3
t6

A readers-writers problem & solution

- *n* processes may access a file for reading or writing.
- They want to coordinate their accesses such that
  - **several processes may read** at the same time;
  - while **one process writes**, **no-one else** may have **access**.

→ Problem as PT-system shown in **black**
(parameterized scheme, for any *n*)

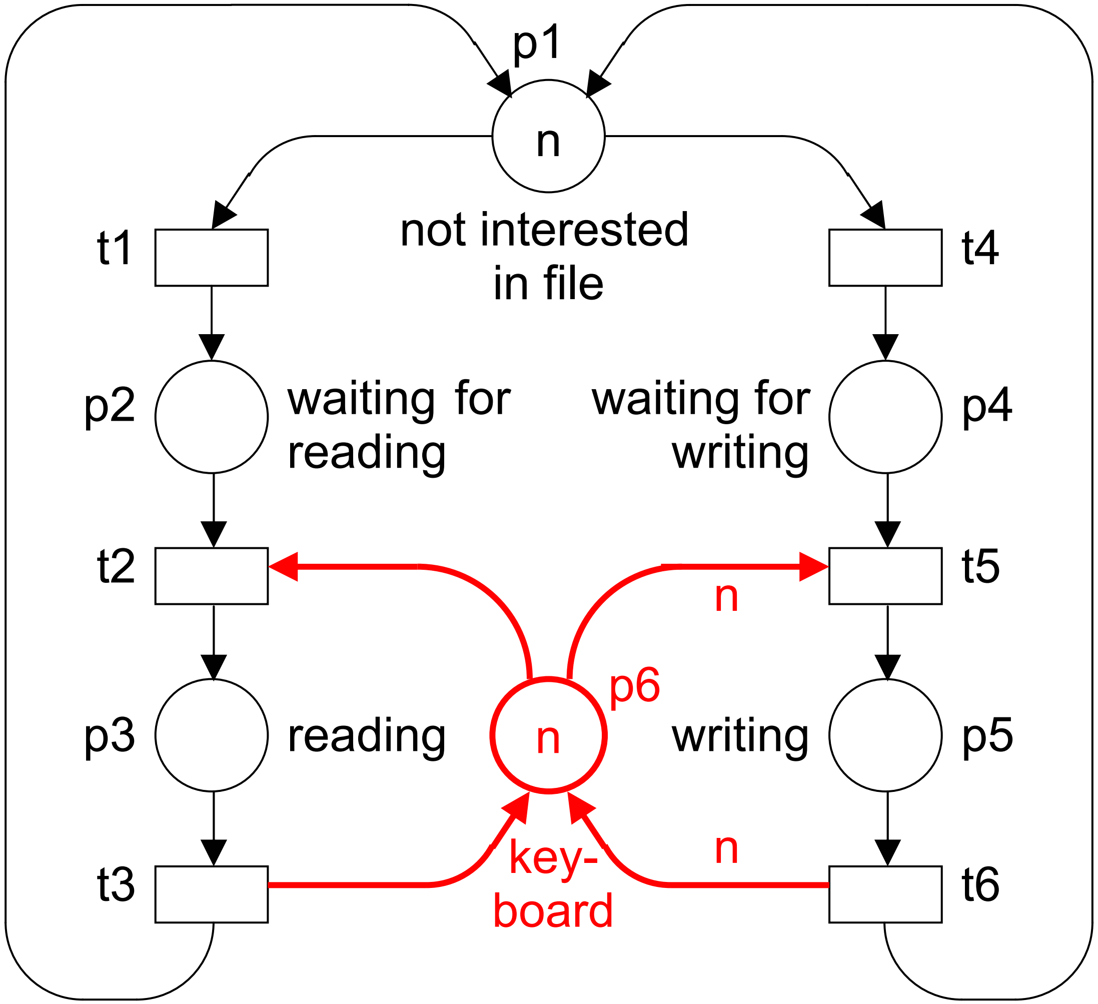
Solution idea: ***n*-keys algorithm**

A keyboard holds *n* keys.

A reader takes 1 key before reading and returns it afterwards.

A writer needs *n* keys.

Solution net



Correctness: Linear analysis plus …

Incidence matrix:

$$C = \begin{pmatrix} -1 & 0 & 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 1 & -1 & 0 & -n & n \end{pmatrix}$$

Two P-invariants:

$$C^T \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad C^T \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ n \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

First:

No transition changes the total number of processes.

Second:

#(readers) + n•#(writers) + #(free keys) = n

The second P-invariant implies correctness! (why?)

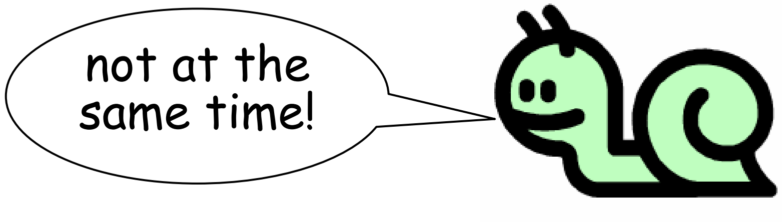
Petri-Nets for concurrency

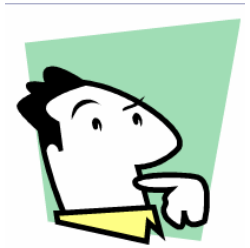
It is hard to really **observe concurrency**.

It is possible to observe effects of causality
(often → non-simultaneity).

It is possible to obtain local timed observation sequences
(often → **potential** simultaneity)

Relativity → There is no **absolute** simultaneity of time-point events!



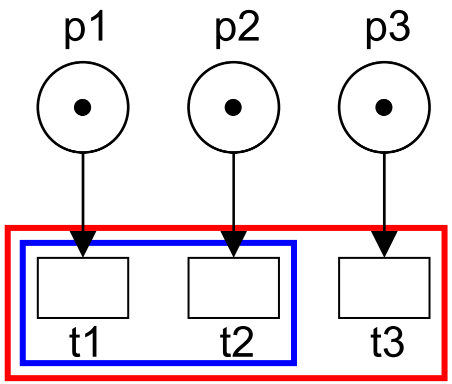


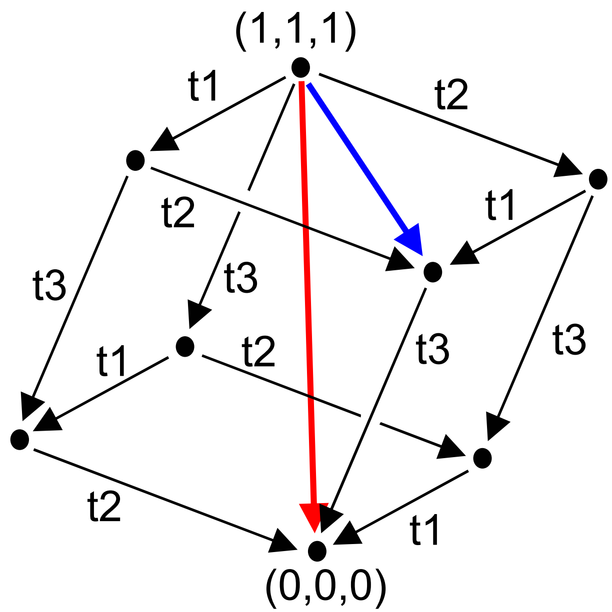
Therefore: I prefer to consider events as **concurrent** which

- are not mutually exclusive (e.g. because they use a resource – place, tool, permission – that exists only once) and
- can be observed to happen in any order, arbitrarily closely together.

Concurrency in the reachability graph …

- does not produce new system states (markings);
- produces new arcs
  which are **diagonals** of
  n-dimensional cubes (n=2, 3, 4, …)
  in the reachability graph:





Some Literature

B. Baumgarten: *Petri-Netze, Grundlagen und Anwendungen* (2nd ed.), Spektrum Akademischer Verlag, 1996

M. Diaz: *Les réseaux de Petri, Modèles fondamenteaux*, Hermes Science, 2001

T. Murata: Petri nets. *Proceedings IEEE*, 77/14, 541-579, 1989

Wolfgang Reisig: *Petri Nets, An Introduction*. EATCS Monographs on Theoretical Computer Science, Springer, 1985

P.H Starke : Analyse von Petri-Netz-Modellen, Teubner, 1990

→ Petri Nets World website
http://www.informatik.uni-hamburg.de/TGI/PetriNets/