Introduction to Model-Checking

Theory and Practice

Beihang International Summer School 2019

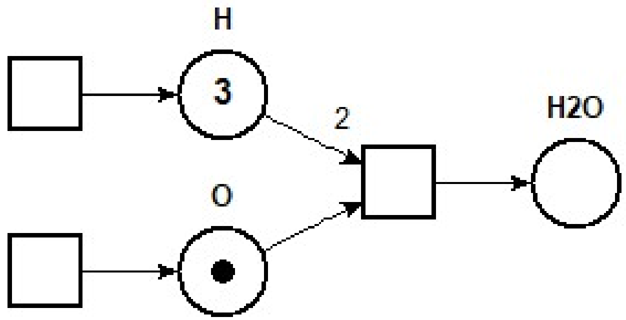
http://homepages.laas.fr/dalzilio/courses/mccourse

Petri Nets

a model for concurrency

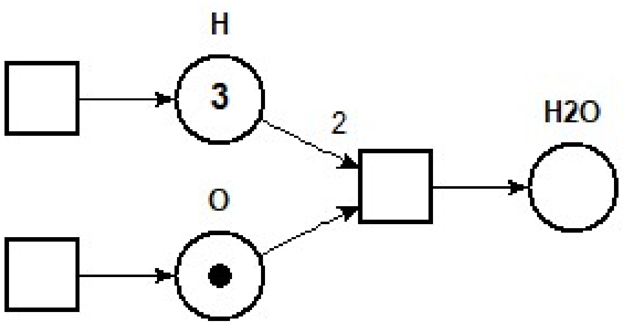
Petri Nets

- Petri nets are a basic model of parallel and distributed systems, designed by Carl Adam Petri in 1962 in his PhD Thesis: “Kommunikation mit Automaten”
- The basic idea is to describe *state changes* in a system using transitions



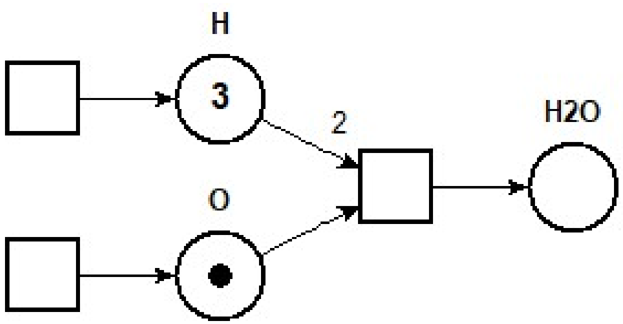
Petri Nets

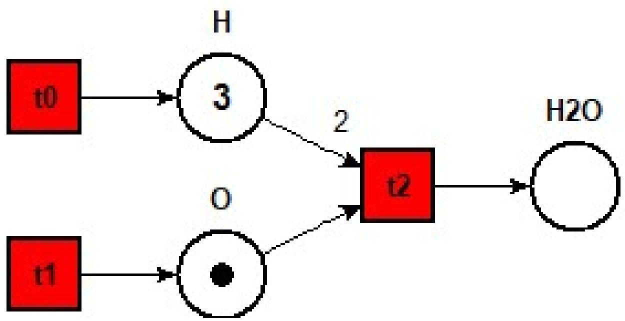
- Petri nets contain places (circle) and transitions (square) connected by directed arcs.
- Transitions (T) ≡ actions
- Places (P) ≡ states or conditions that need to be met before an action can be carried out.
- Places may contain tokens that move when transitions fire.



Petri Nets

- Places may contain tokens that move when transitions fire.

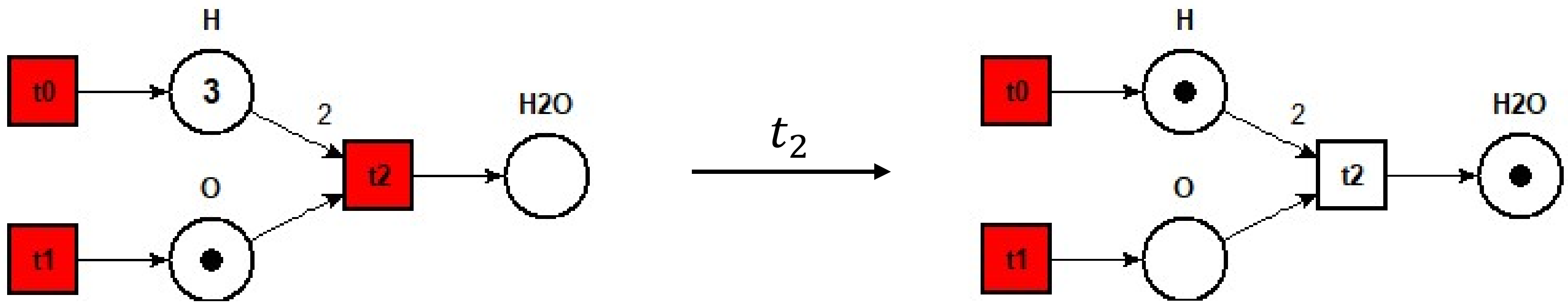




enabled transitions

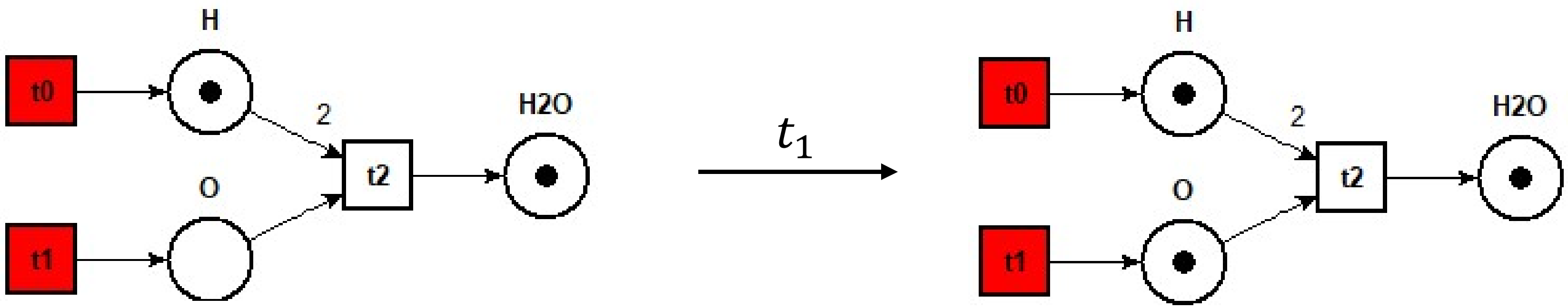
Petri Nets: the token game

- Places may contain tokens that move when transitions fire.



Petri Nets: the token game

- Places may contain tokens that move when transitions fire.



open

Petri Nets

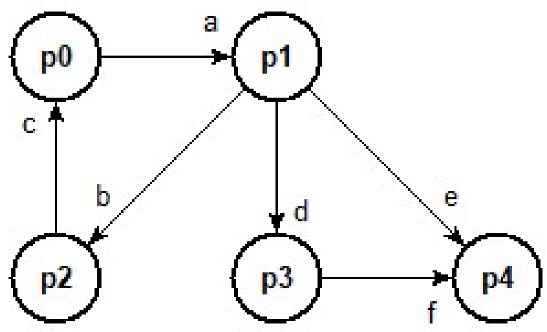
Some Examples

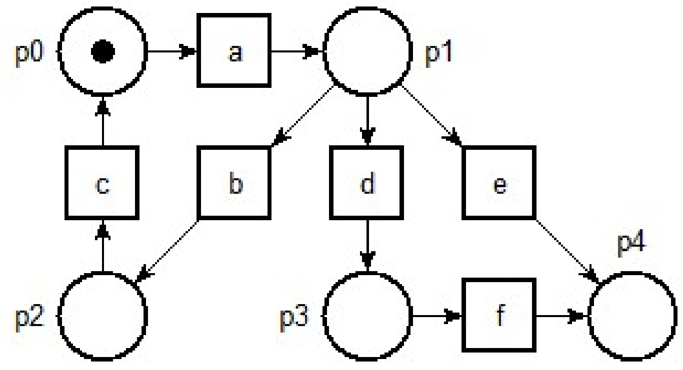
The tool Tina

- All the examples and exercises in this course will make use of Tina, a toolbox for the model-checking of time Petri net
- Download at:

http://projects.laas.fr/tina/

Automata as Petri nets

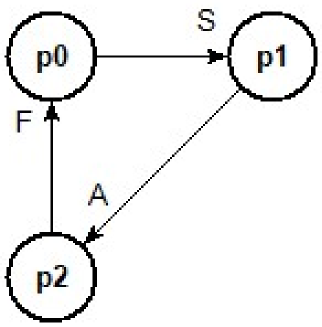


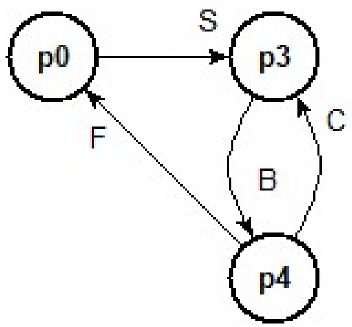


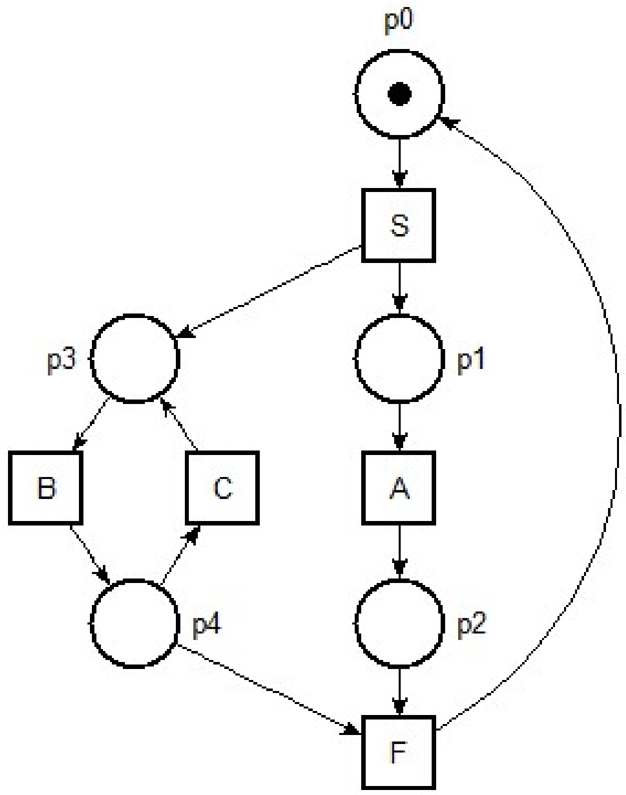
a. b. c. a. d. f. 死

open

Synchronizing Automata

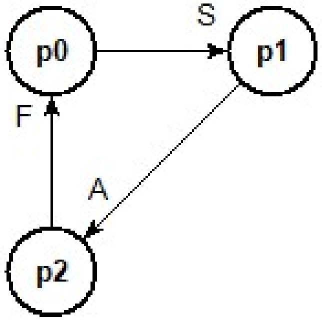






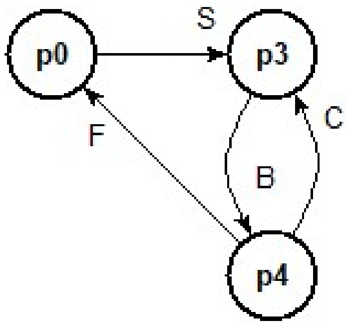
open

Synchronizing Automata

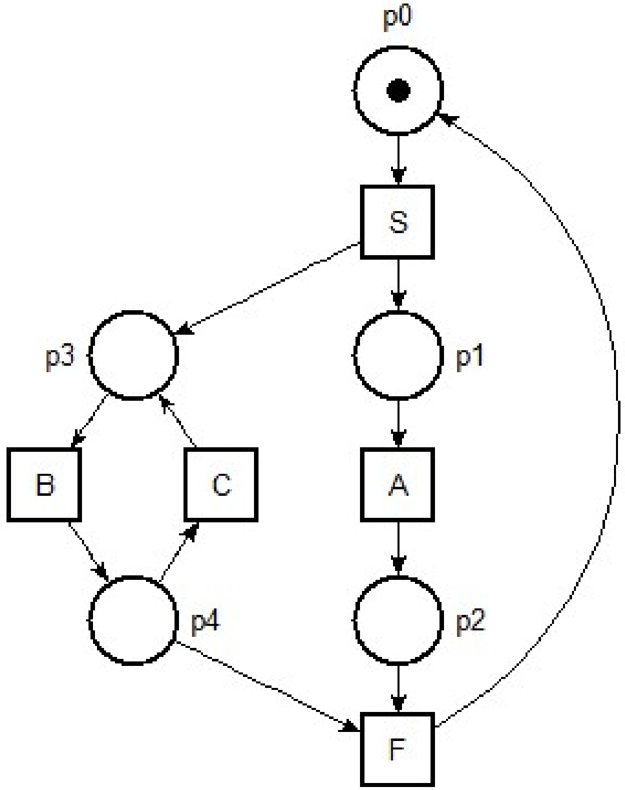


S.A.F

S		A			F
=	∉	∉	∉	∉	=
S	B		C	B	F



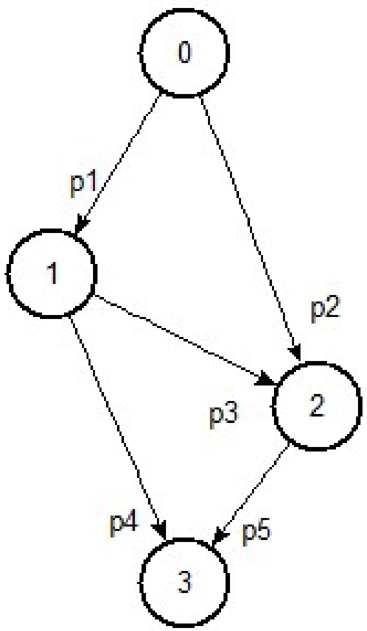
S.B.C.B.F

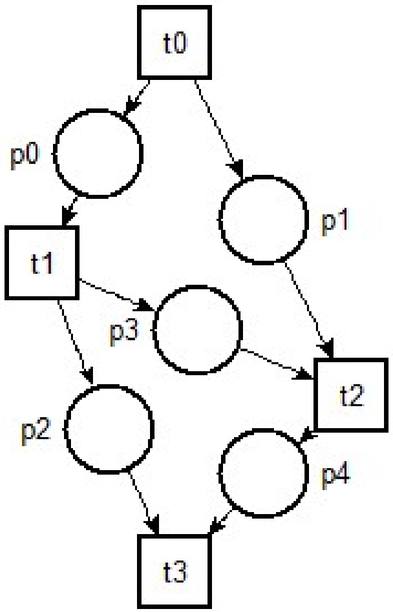


S.B.A.C.B.F

(B.A.C.B) is in the *shuffle* of A and B.C.B

Graph of Tasks



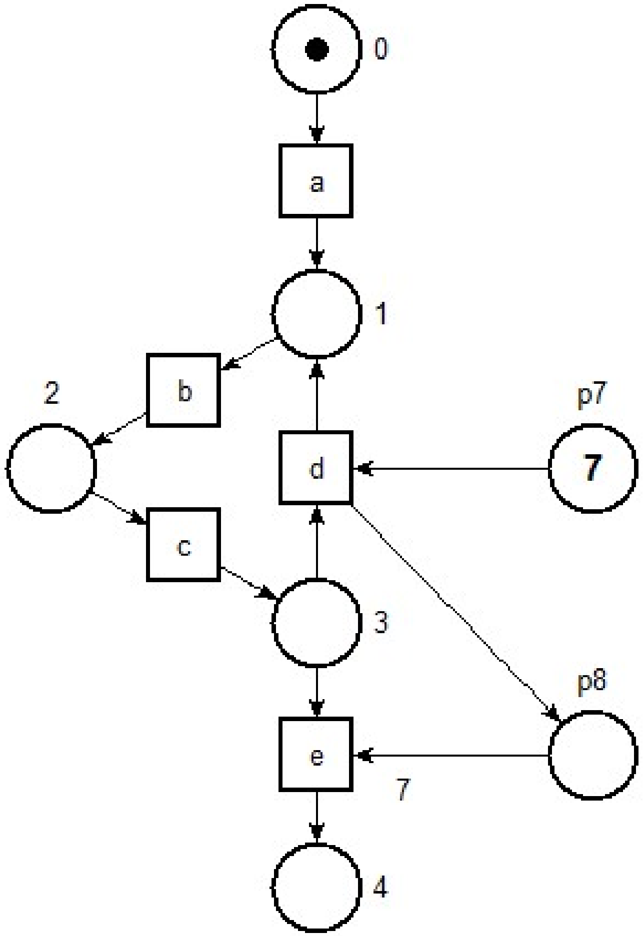


$$P_1 \leq P_3 \land P_1 \leq P_4$$

$$P_2 \leq P_5 \land P_3 \leq P_5$$

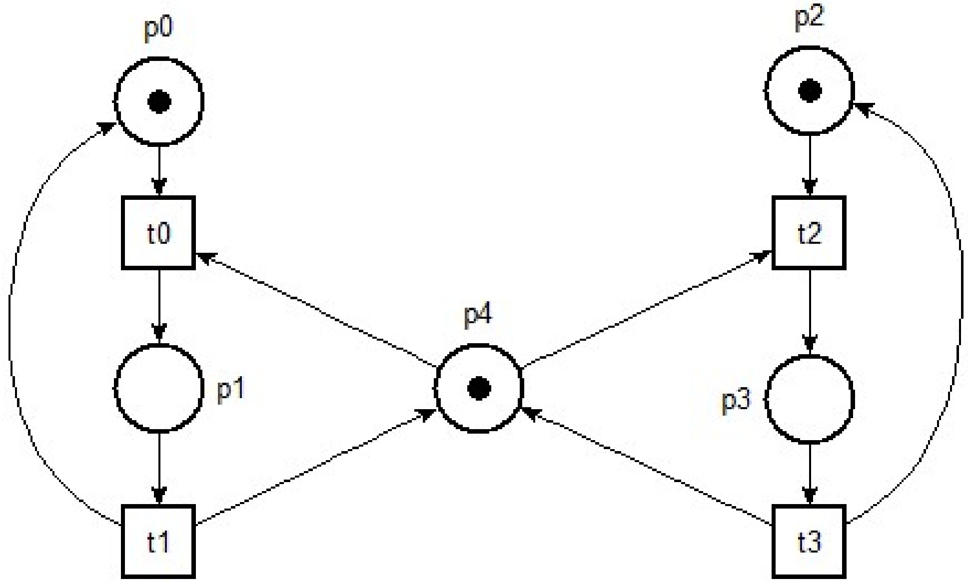
open

More examples: counters



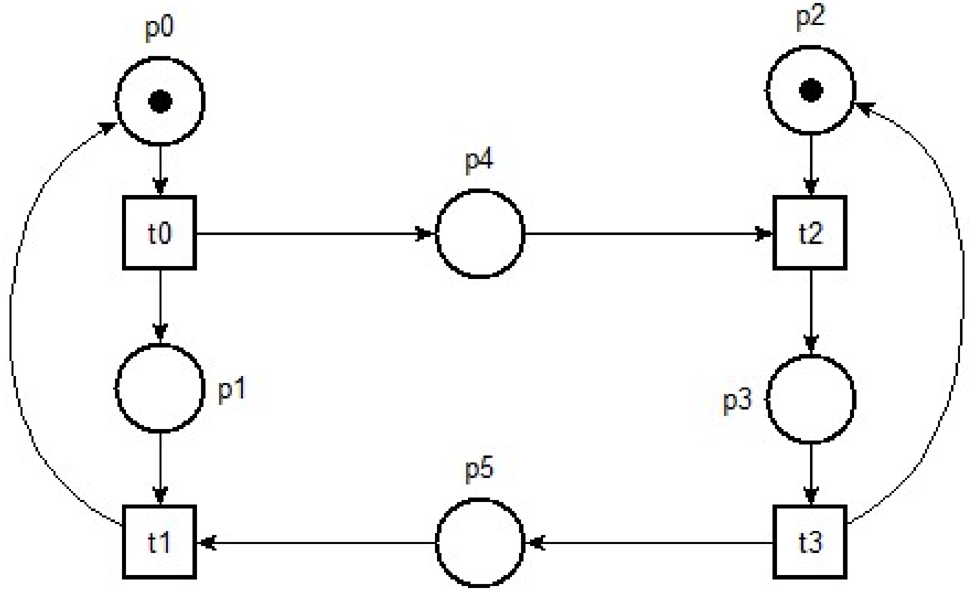
open

More examples: Mutual Exclusion



open

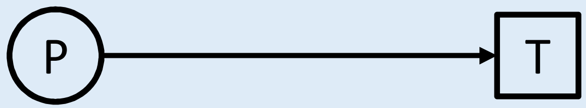
More examples: Message Passing



open

What you need to remember

- A net has places and transitions (nets are static)
- Tokens gives the current state of the net (markings)
- Arcs are conditions for a transition to fire; they model the flow of interaction

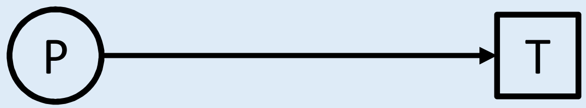


P ≡ passive component
stores tokens:
resources; data;
buffers; locks

T ≡ active component
resource consumption ;
data changed ;
lock acquired

What you need to remember

- A net has places and transitions (nets are static)
- Tokens gives the current state of the net (markings)
- Arcs are conditions for a transition to fire; they model the flow of interaction



arcs ≡ flow

physical proximity ;

data access right ;

network topology

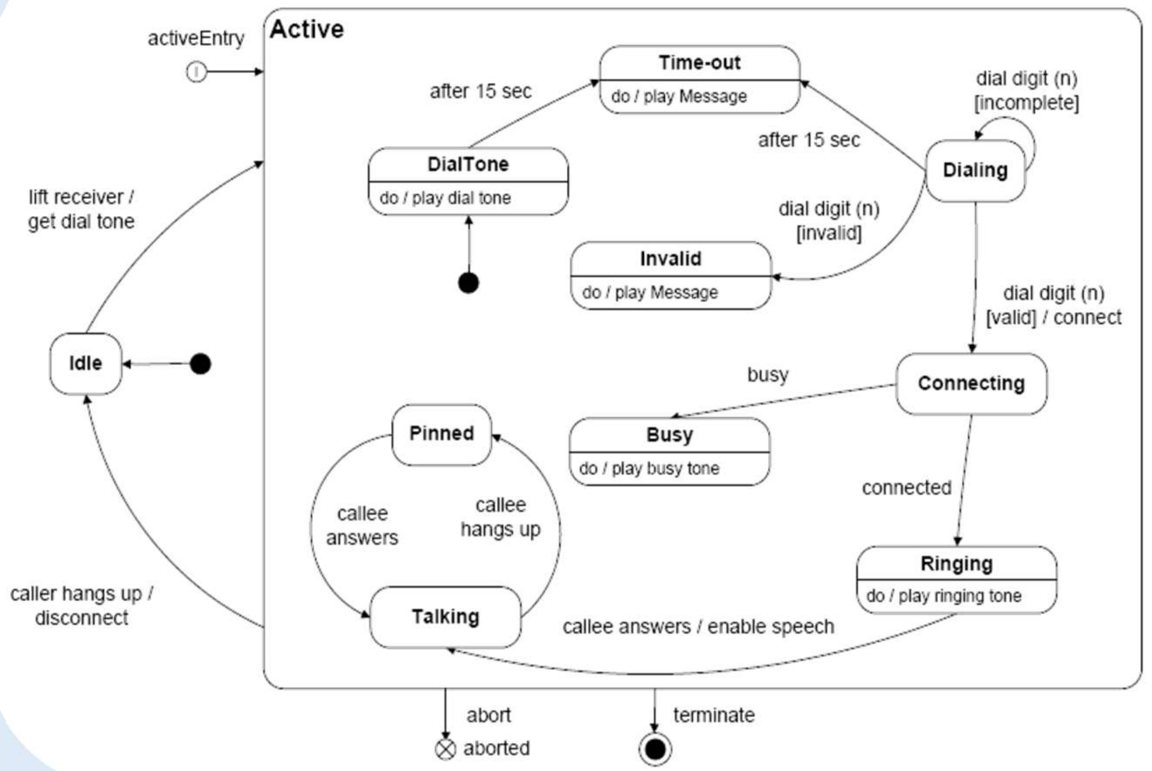
What you need to remember

With the same formalism we can model

- *concurrency*: transitions may fire independently (see synchronizing automata example)
- *causality*: firing transitions depends on the current state (see mutual exclusion and PER tasks examples)
- *resources*: see the counters example
- *global state*: state is distributed over places
- *compositionality* (or *component-based modeling*)

We have a single, unified way to model states, data, computations and synchronization

Using Diagrams (e.g. Statecharts)



M. L. Crane, J. Dingel (2005). UML vs. Classical vs. Rhapsody Statecharts: Not All Models Are Created Equal. *Int. Conf. on Model Driven Engineering Languages and Systems*

Statecharts may have ≠ interpretations in ≠ tools:

UML statecharts ≠ Classical statecharts ≠ Rhapsody statecharts

Place/Transition Nets

a model for concurrency

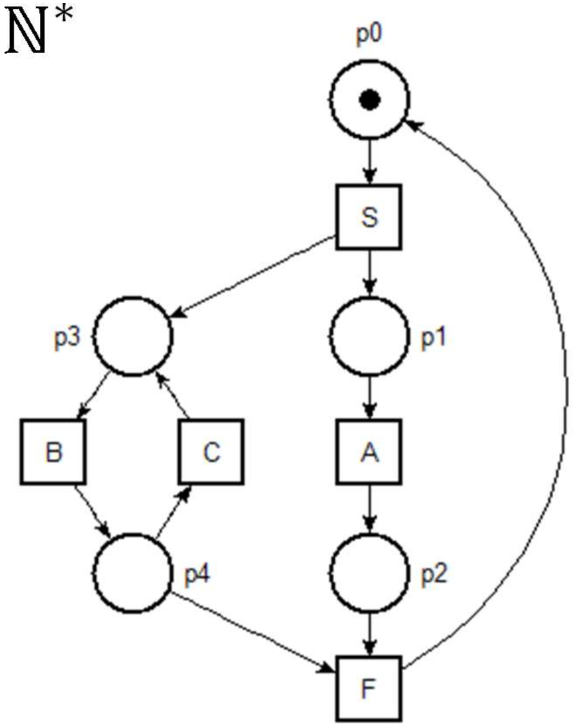
P/T Nets

A P/T net is a tuple $N = \langle P, T, F, W \rangle$ where

- P is a finite set of places
- T is a distinct finite set of transitions ($P \cap T = \emptyset$)
- F is the flow relation: $F \subseteq (P \times T) \cup (T \times P)$
- W are the weight of the arcs: $W : F \rightarrow \mathbb{N}^*$

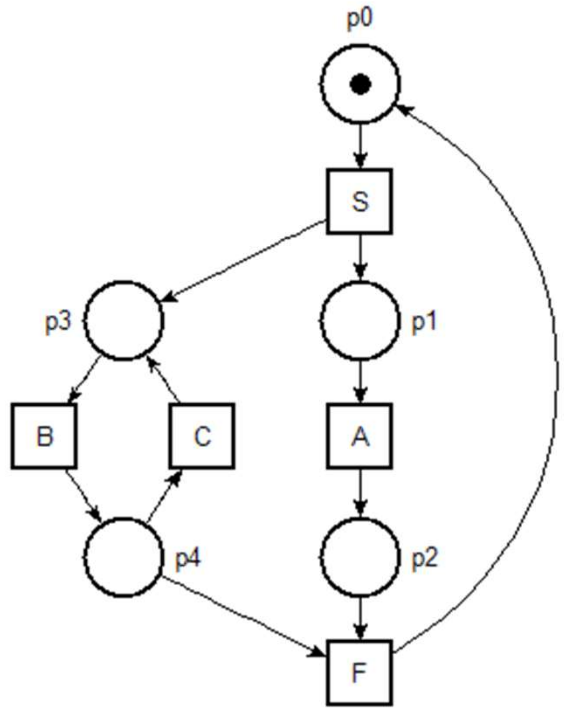
A marking m defines a distribution of tokens to places $m : P \rightarrow \mathbb{N}$

A marked P/T net (N, m_0) is a net with initial marking m_0



P/T Nets

- $P = \{p_0, p_1, p_2, p_3, p_4\}$
- $T = \{S, A, B, C, F\}$
- $F = \{(p_0, S), (S, p_1), (S, p_3), \ldots\}$
- all weights are 1 (it is an *ordinary net*)



$m = \{p_0 : 1, p_1 : 0, p_2 : 0, p_3 : 0, p_4 : 0\}$

$m = \{p_0\}$

Notations

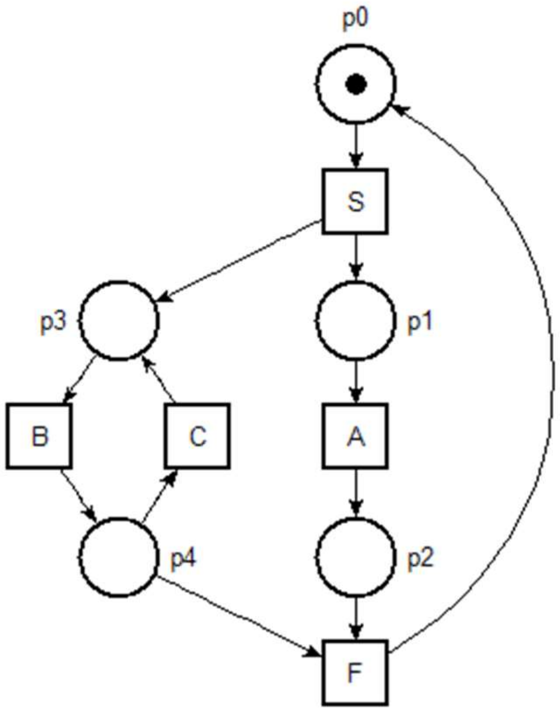
- If $(p, t) \in F$ then p is an input place of t
- If $(t, p) \in F$ then p is an output place of t

- The set $Pre(p) = \{t \mid (t, p) \in F\}$ is the pre-set of p (same with $Pre(t)$)

  $Pre(F) = \{\mathrm{p}_2, \mathrm{p}_4\}$

- The set $Post(p) = \{t \mid (p, t) \in F\}$ is the post-set of p

  $Post(p_4) = \{\mathrm{C}, \mathrm{F}\}$

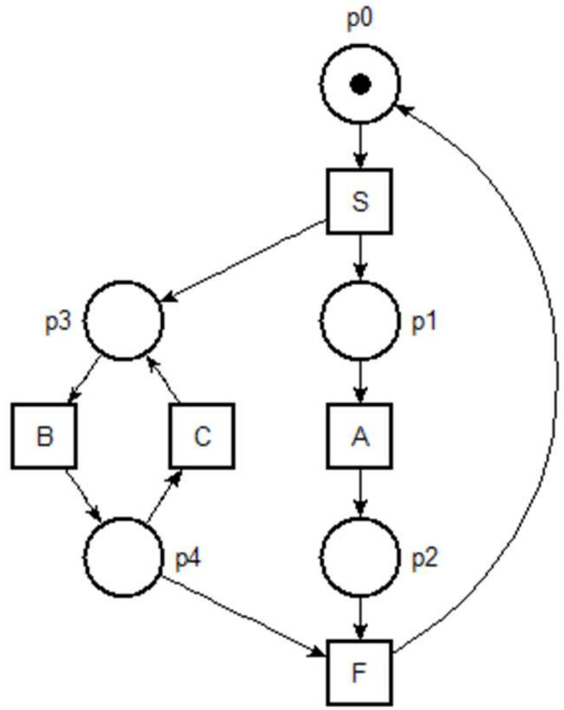


By extension we write

$Pre_t(p) = W(p,t)$ if $p \in Pre(t)$
and $Pre_t(p) = 0$ otherwise

$Post_t(p) = W(t,p)$ if $p \in Post(t)$
and $Post_t(p) = 0$ otherwise

$$\begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} \qquad Pre_F = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \qquad Post_F = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



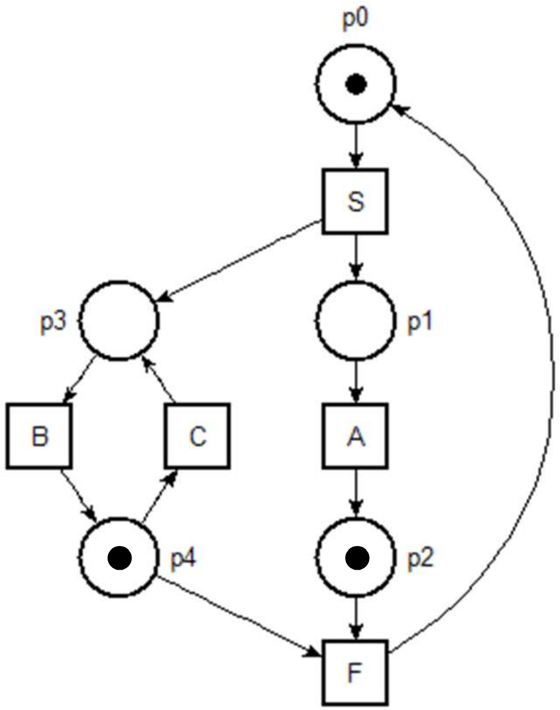
Firing condition (enabledness)

transition $t \in T$ is enabled on the marking m, written $m \rightarrow^{t}$, iff $\forall p \in Pre(t).\,(m(p) \geq W(p,t) \geq 0)$

or equivalently: $m \ - Pre_t \geq \bar{0}$

e.g. F is enabled on $m = \{p_0, p_2, p_4\}$

$$m = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \geq Pre_F = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

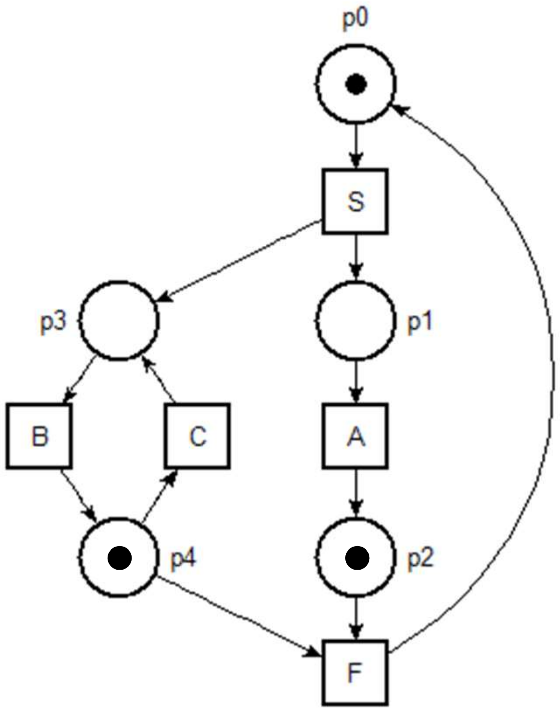


Firing rule

if $t \in T$ is m-enabled then t can fire and produces the marking m', written $m \rightarrow^{t} m'$, such that:

$$\forall p \in P.\, (m'(p) = m(p) - Pre_t(p) + Post_t(p))$$

i.e. $m' = m - Pre_t + Post_t$



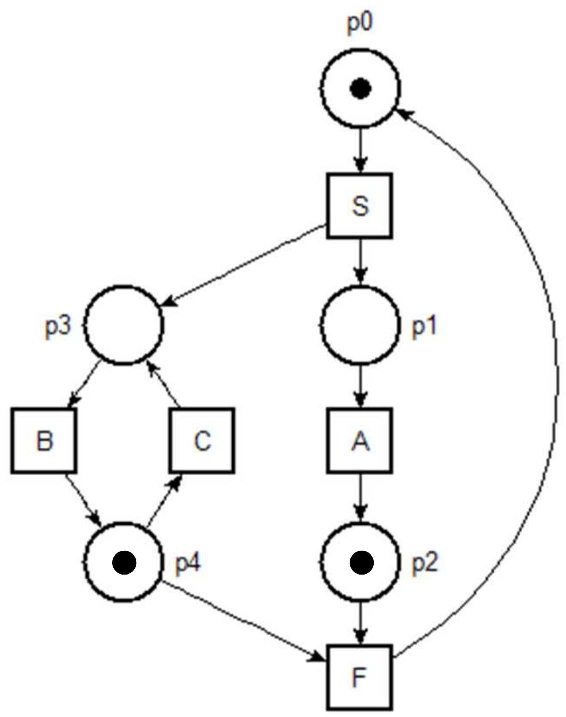
Firing transition F from m

if $t \in T$ is m-enabled then t can fire and produces the marking m', written $m \rightarrow^t m'$, such that:

$$\forall p \in P.\left(m'(p) = m(p) - Pre_t(p) + Post_t(p)\right)$$

$$m' = m - Pre_F + Post_F$$

$$m' = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



Remark

- It is possible to express most of the results on Petri nets using linear algebra (see later) → see also the VASS model (Vector Addition System with States).

$$a(t_i, p_j) = W(t_i, p_j) - W(p_j, t_i)$$

$$N = \begin{bmatrix} a(t_1, p_1) & \cdots & a(t_n, p_1) \\ \vdots & \ddots & \vdots \\ a(t_1, p_k) & \cdots & a(t_n, p_k) \end{bmatrix} \text{ and } m'' = m' + N \times \begin{bmatrix} 0 \\ \cdots \\ 1 \\ \cdots \end{bmatrix}^T$$

- Beware! the positivity constraint in the firing condition, $m - Pre_t \geq \bar{0}$, makes everything harder.

Reachability Graph

Reachable Markings

Let m be a marking of the marked net (N, m_0) with $N = \langle P, T, Pre, Post \rangle$.

The set of markings reachable from m (the reachability set of m) is the smallest set $reach(m)$ such that:

1. $m \in reach(m)$
2. $m' \in reach(m) \land m' \rightarrow^{t} m'' \Rightarrow m'' \in reach(m)$

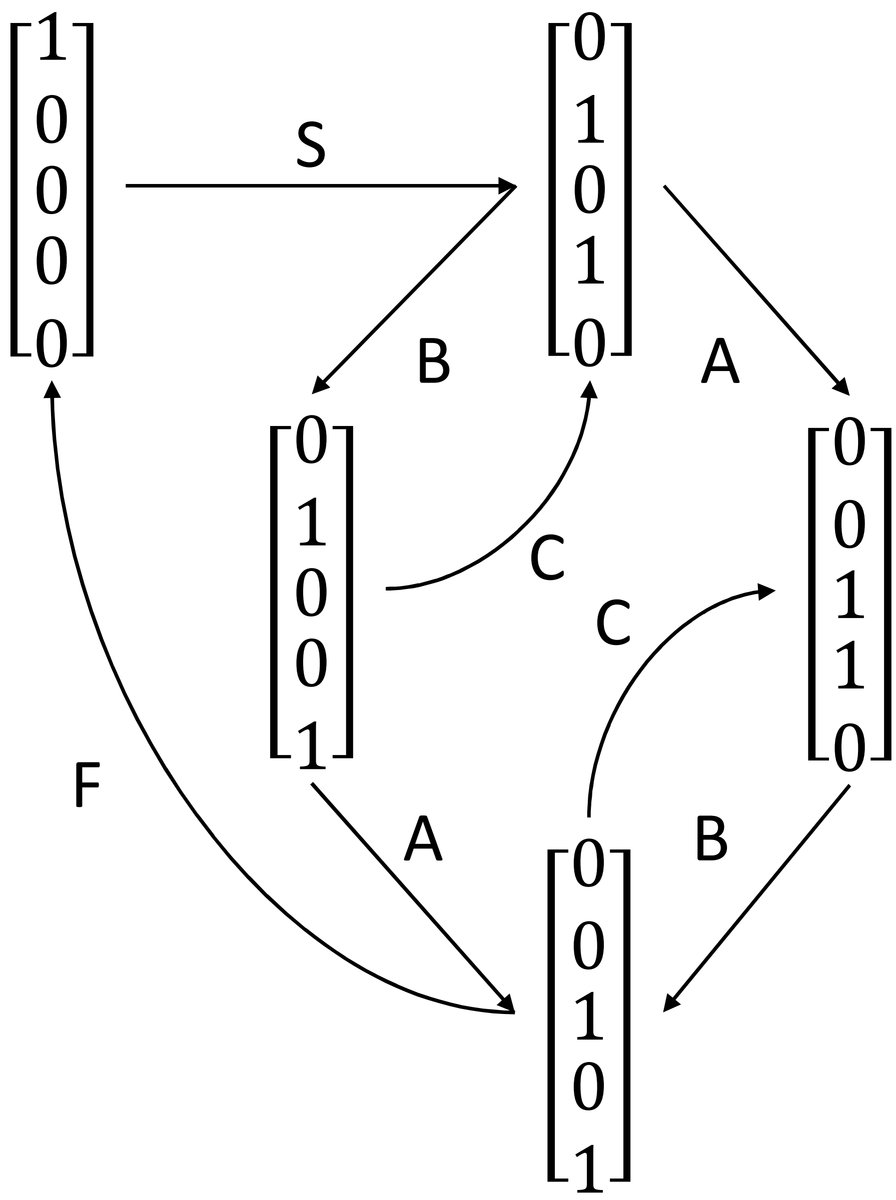
Reachability Graph

The reachability set of a (marked) net is the set $reach(m_0)$

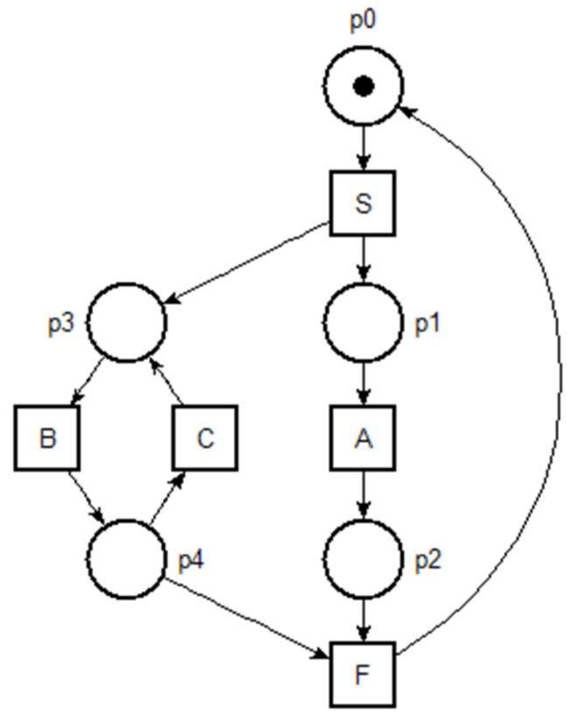
The reachability set is not necessarily finite

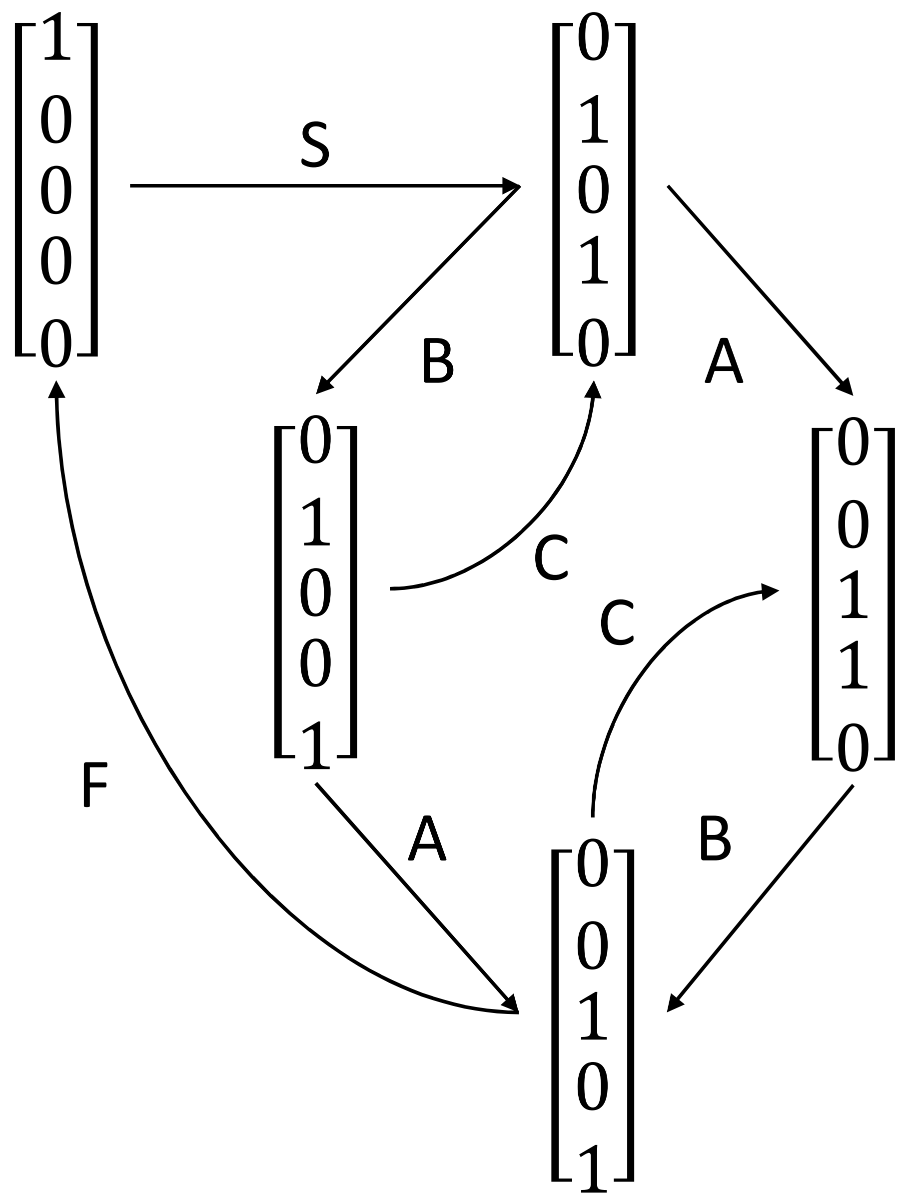
The reachability graph of a net is the rooted graph (V, E) such that:

1. $\mathrm{V} = reach(m_0)$ and the root is $v_0 = m_0$
2. $(m_1, t, m_2) \in E$ iff $m_1 \to^t m_2$

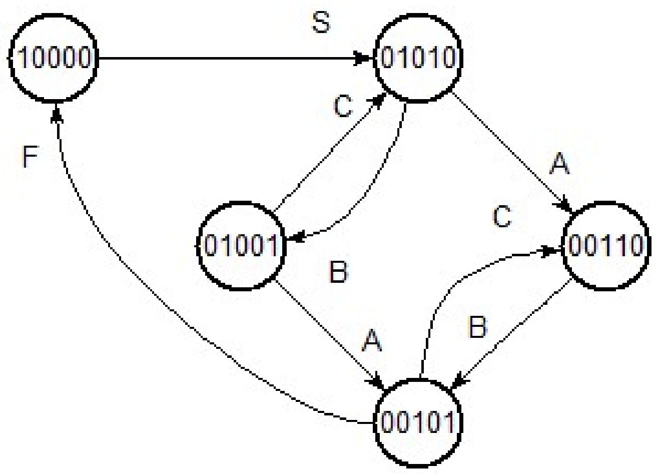


$$\begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

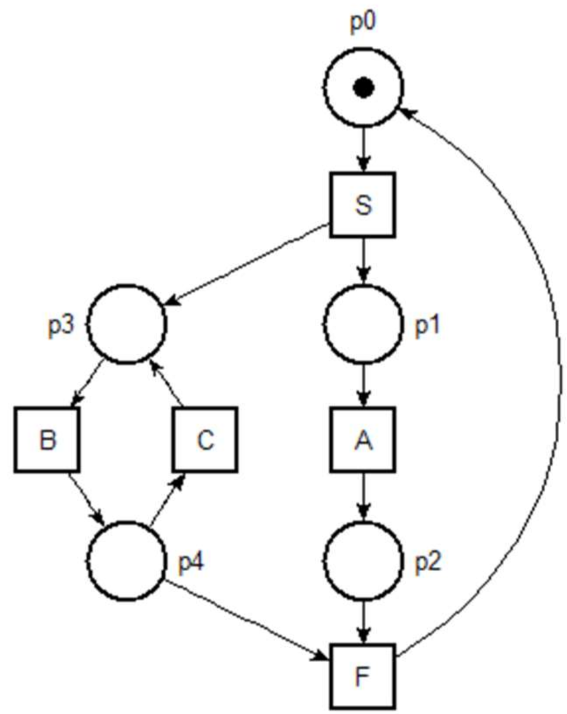




S
B
A
C
C
A
B
F

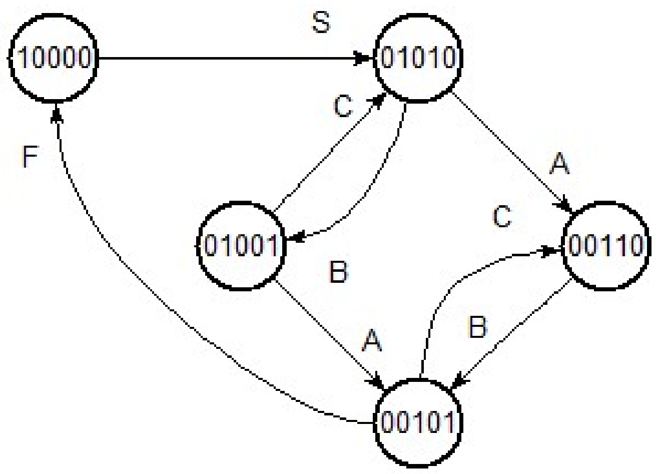


10000
01010
01001
00110
00101
S
C
A
B
C
A
B
F



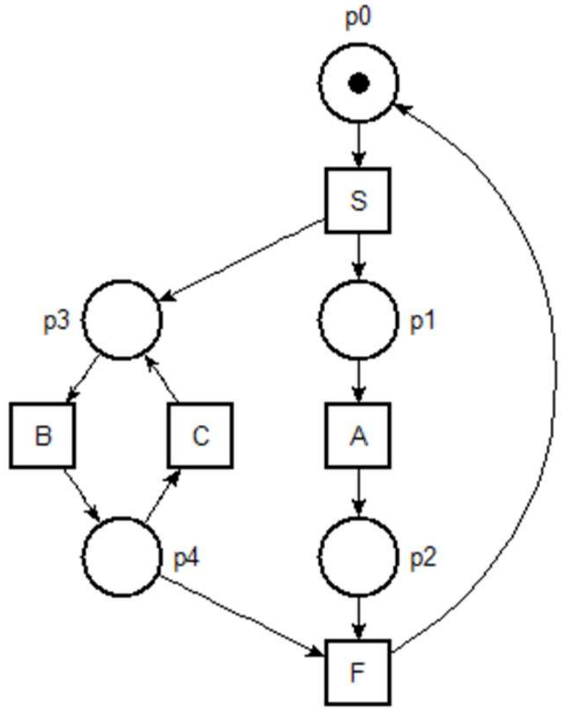
p0
S
p3
p1
B
C
A
p4
p2
F

Occurrence Sequence



Labels of the transitions along a path starting at m_0

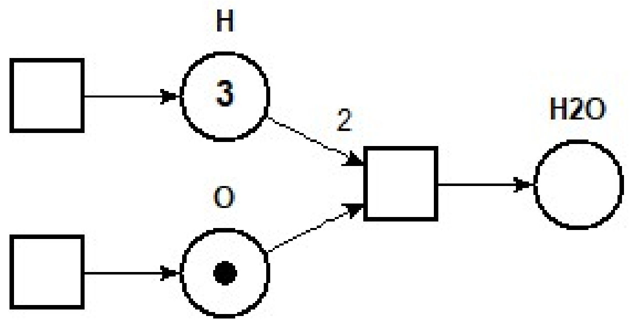
e.g. ϵ, S.A.B, S.B.A.F, …



Equality of language provides a nice notion of *equivalence*

Size of the Reachability Graph

- The graph may be infinite if there is no bound on the number of tokens in a place.



- If each reachable marking can contain at most k tokens in each place then the (marked) net is said to be *k-safe.*
- A k-safe net has at most $(k+1)^{|\mathrm{P}|}$ markings.

What you need to remember

- Marking (reachability) graph provides a way to explain the behavior of a net. We call this its semantics.

  This is the central tool to talk about verification

- The "graph" is deterministic ($\neq$ transitions have $\neq$ names). This is not necessarily true if you work with labeled nets.

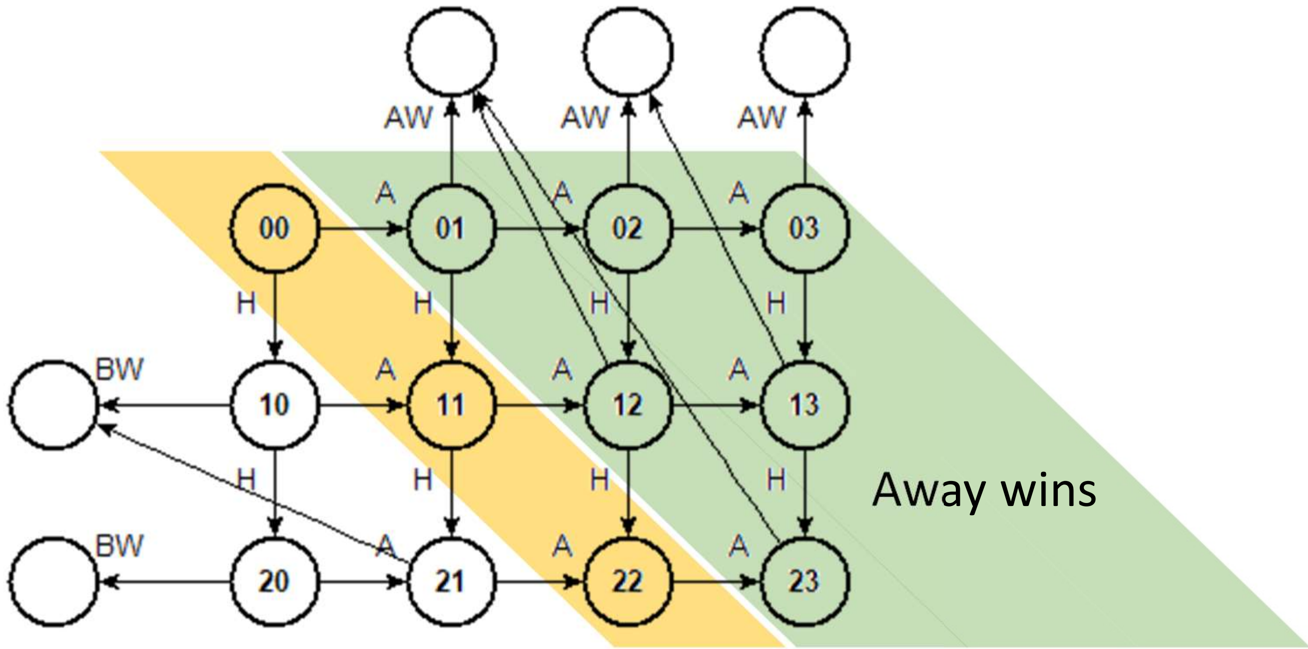
- Reachability graph may be encountered in many area of formal verification ($\approx$ Kripke structures).

Petri Nets

Coming back to one of our examples

Soccer Game

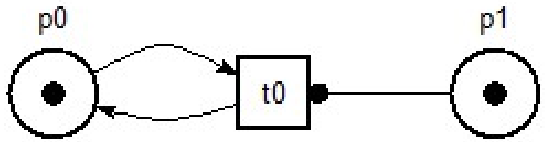
- Remember the soccer game example ? Try to model it with a Petri net.



$A.H.A.\dots.H.H.AW \in \mathcal{L}$ if there are more A than H

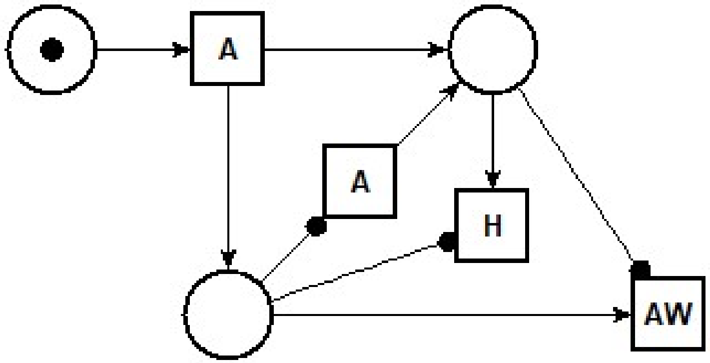
Petri nets: extended arcs

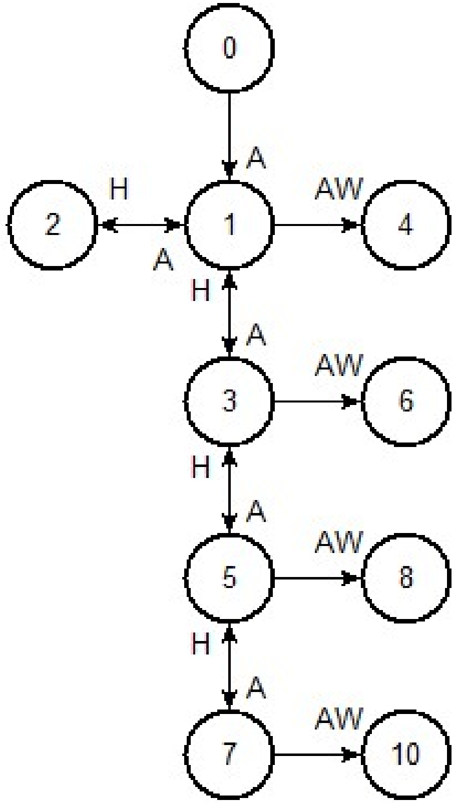
- Read arcs: check whether the place is marked



- This only affects *enabledness* (firability); the marking of p_1 does not change when t_0 fires
- This is the same has taking a token and putting it back! → we say that *there is no gain in expressive power*

Soccer game: ½-solution ?



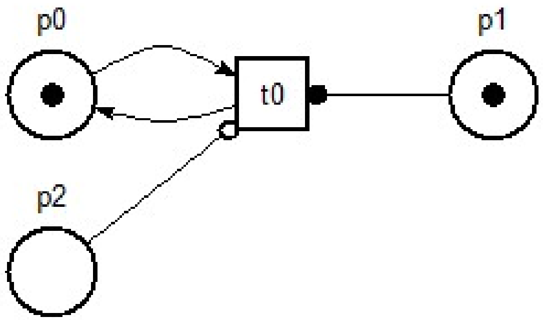


open

Away team wins

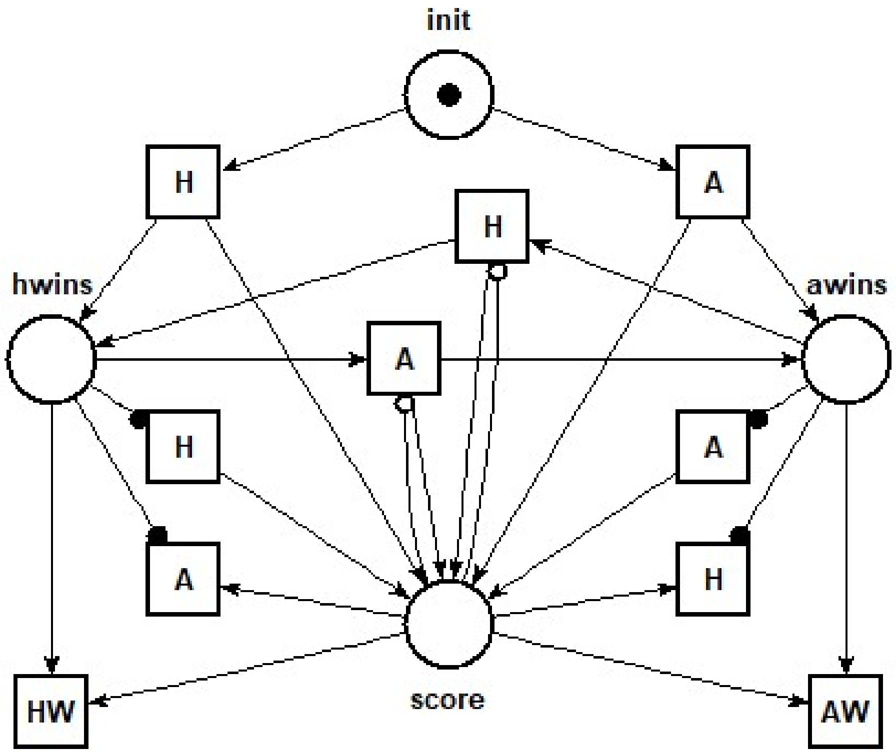
Petri nets: extended arcs

- Inhibitor arcs: constrain a place to be empty



- Used to test if the marking is zero

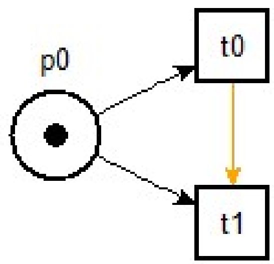
Soccer game with inhibitor arcs



open

Petri nets: extended arcs

- Priorities: prevent a transition from firing if another one can (here t_0 can fire but never t_1)



- You can also find *flush arcs* (empty a place of its tokens); *test arcs*; *transfer arcs; …*

What you need to remember

- Every finite state graph can be “modeled” with a Petri; even if this is not necessarily a good choice
- There are examples of systems that cannot be modeled with Petri nets
- Extensions are useful but they may have a cost

Some theoretical results on P/T nets

Complexity theory for P/T net

- All interesting questions about the behavior of 1-safe Petri nets are PSPACE-hard (so may require exponential time).
  - reachability, liveness,
- Equivalence problems for 1-safe nets may require exponential space.
- All interesting questions about the behavior of general Petri nets are EXPSPACE-hard (and require at least $2^{O(\sqrt{n})}$-space), and equivalence problems are undecidable

Javier Esparza, Petr Jančar, and Alexander Miller. 2008. On the Complexity of Consistency and Complete State Coding for Signal Transition Graphs. *Fundam. Inf.* 86, 3 (2008).

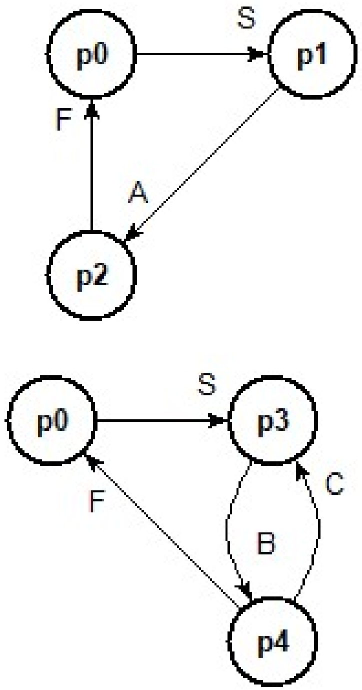
Reachability

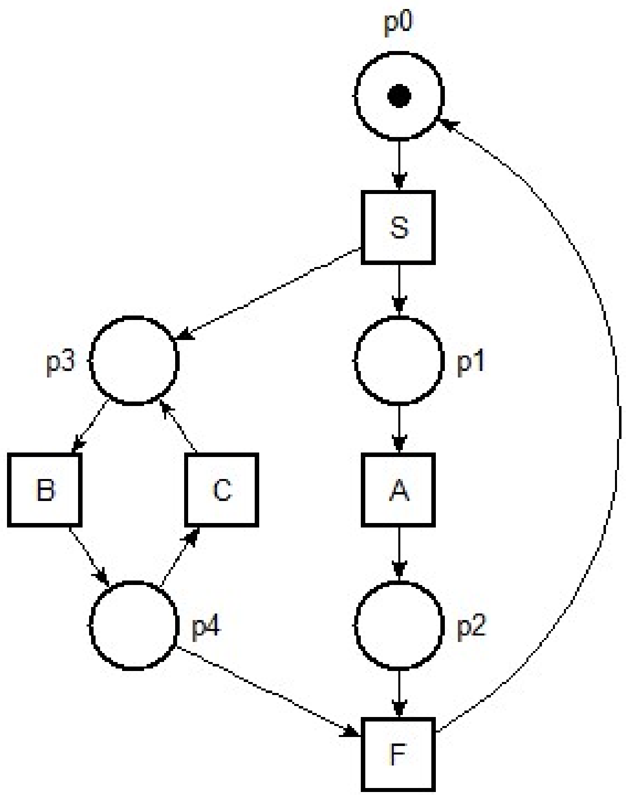
- In the general case, the reachability problem was shown to be decidable by Mayr and shortly after, with a simpler (!?) proof, by Kosaraju
- The problem is at least EXPSPACE-hard
- All known, complete algorithm are non-primitive recursive
- The problem becomes undecidable with nets that have (at least 3) inhibitor arcs.

Composition of Nets

Product of automata

- Remember the synchronizing automaton example?
- A similar operation can be done directly on graph using a *product operation*



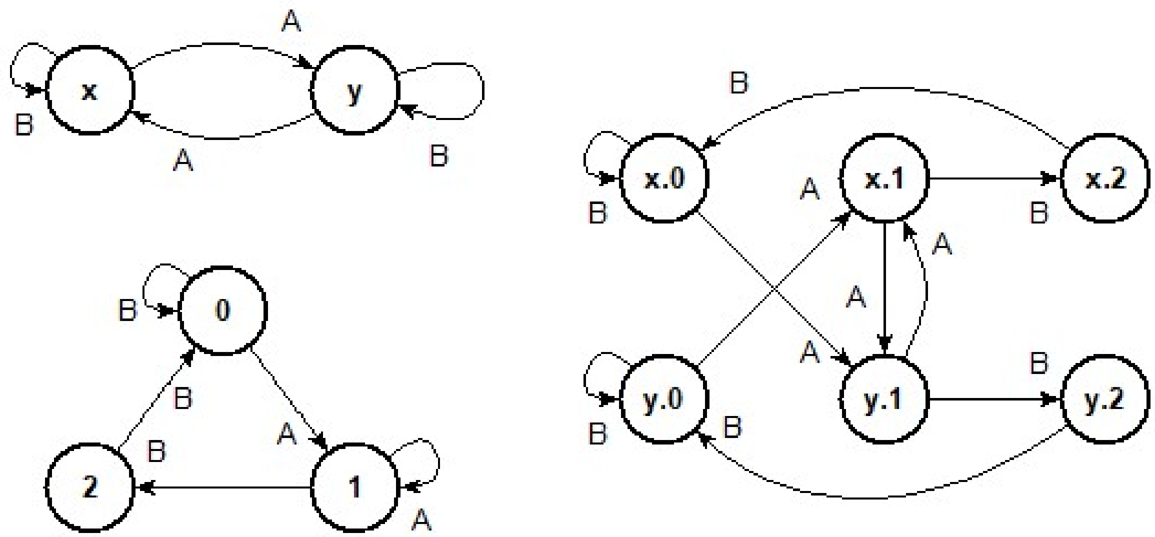


Product of automata: $\mathcal{A}_1 \otimes \mathcal{A}_2$

- Imagine that we have some (product) operation $\otimes$ on the labels of automata
- From two automata $\mathcal{A}_1 = (Q_1, \Delta_1, q_0^1)$ and $\mathcal{A}_2 = (Q_2, \Delta_2, q_0^2)$ we can define their product $\mathcal{A}_1 \otimes \mathcal{A}_2$ has the automata with states in $Q_1 \times Q_2$ (cartesian product) and initial state (q_0^1, q_0^2)
- We have several possibility for defining the "product" transitions.

Example: intersection

We can take transitions that are available on both sides, i.e. $(q_1, q_2) \xrightarrow{a} (q_1', q_2')$ when both $q_1 \xrightarrow{a} q_1'$ and $q_2 \xrightarrow{a} q_2'$

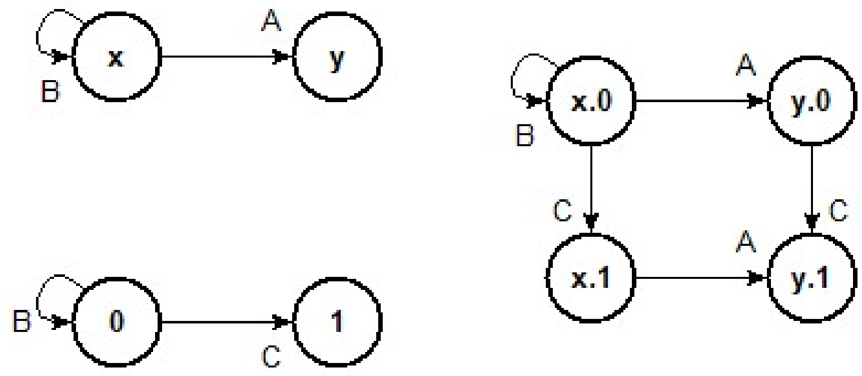


Example: union

We can take transitions that are available only on one side:

$$(q_1, q_2) \xrightarrow{a} (q_1', q_2) \text{ when } q_1 \xrightarrow{a} q_1'$$

and $$(q_1, q_2) \xrightarrow{a} (q_1, q_2') \text{ when } q_2 \xrightarrow{a} q_2'$$



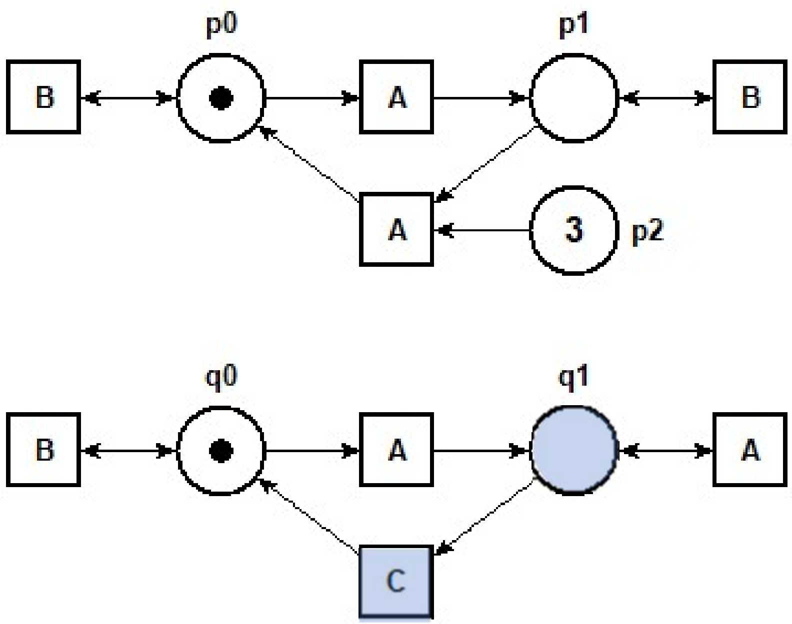
Product of automata

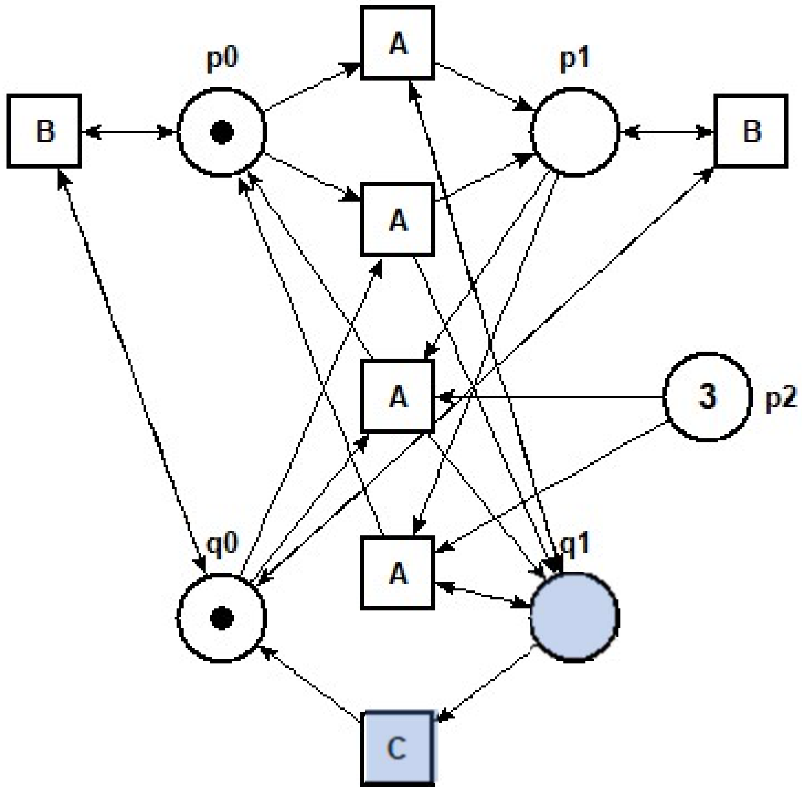
- Likewise we could define the *synchronous product of two automata* or the “shuffle” of two languages
  - shuffle = words obtained by mixing the actions of two words but keeping their relative order (think of a deck of cards)

Product of P/T nets

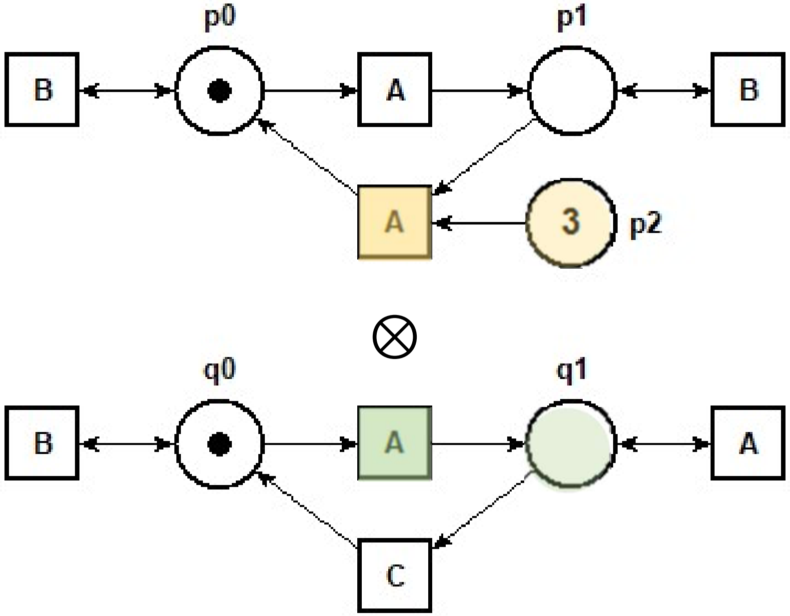
- Given two nets N_1 and N_2 with can define their product in almost the same way.
- This is a net N with places $P = P_1 \cup P_2$
- A transition $t = t_1 \otimes t_2$ is in N iff t_1 and t_2 have the same label. In this case
  - $Pre(t) = Pre(t_1) \cup Pre(t_2)$
  - $Post(t) = Post(t_1) \cup Post(t_2)$
- We can show that the language of N is exactly the synchronous product $\mathcal{L}_1 \otimes \mathcal{L}_2$

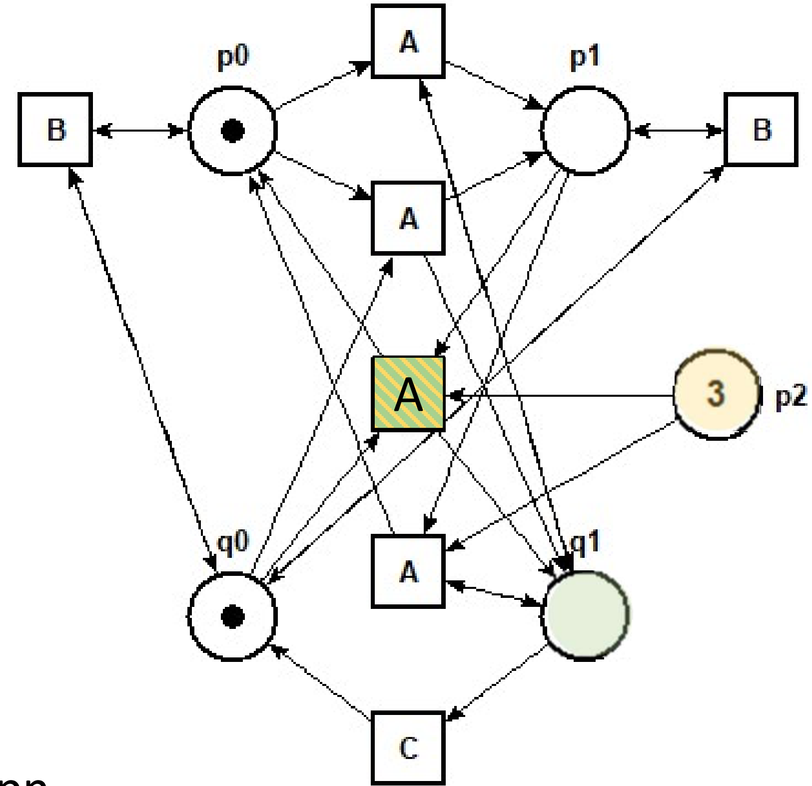
Product of transitions





Product of transitions





script tpn
load A1.ndr
load A2.ndr
sync 2

What you need to remember

- There are natural notion of composition between automata and nets → this is like algebra, where you have a notion of groups $(\mathbb{N}, +, 1, \times, 0)$
- Composition also have an interpretation at the level of the semantics (or the language)