

# A subclass of Petri nets that **allow** simple distributed execution

Arkady Klimov

*Institute of Design Problems in Microelectronics  
Moscow*



*arkady.klimov@gmail.com*

The ruSTEP seminar, Problems Day 30.12.2022

# The Plan

- Petri Nets
- The problem of distributedness and concurrency
- Asynchronous execution, directed nets
- What directed nets are
- Relevant classes of Petri Nets
- FC and S are directed, and what about AC?
- General view of AC net component

# Petri Net, P/T-net

P/T-net is a bipartite directed graph of Places  and Transitions 

P/T-net  $\mathcal{N} = (P, T, F), \quad F \subseteq (P \times T \cup T \times P), \quad P \cap T \neq \emptyset$

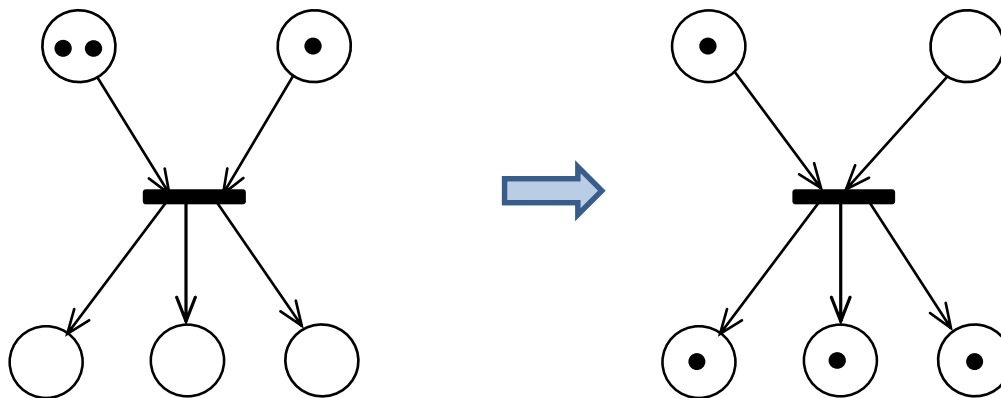
P/T-system  $\Sigma = (\mathcal{N}, M_0), \quad M_0: P \rightarrow \mathbb{N}$  - initial marking

Notation:  $\bullet t = \{p \mid (p, t) \in F\}, \quad t \bullet = \{p \mid (t, p) \in F\}$ , similarly  $\bullet p, p \bullet$

The current state is a marking  $M: P \rightarrow \mathbb{N}$ , where  $\mathbb{N} = \{0, 1, 2, \dots\}$   
(i.e. a number of tokens  $\bullet$  at each place)

Transition  $t$  is enabled in  $M$  if  $\bullet t \leq M$ , where we treat  $\bullet t$  as function  $P \rightarrow \{0, 1\}$

Transition acts as a state transformer:  $M' = M - \bullet t + t \bullet$



Each IN arc captures a token, each OUT arc deposits a token

# Petri Net, P/T-net

P/T-net is a bipartite directed graph of Places  and Transitions 

P/T-net  $\mathcal{N} = (P, T, F), \quad F \subseteq (P \times T \cup T \times P), \quad P \cap T \neq \emptyset$

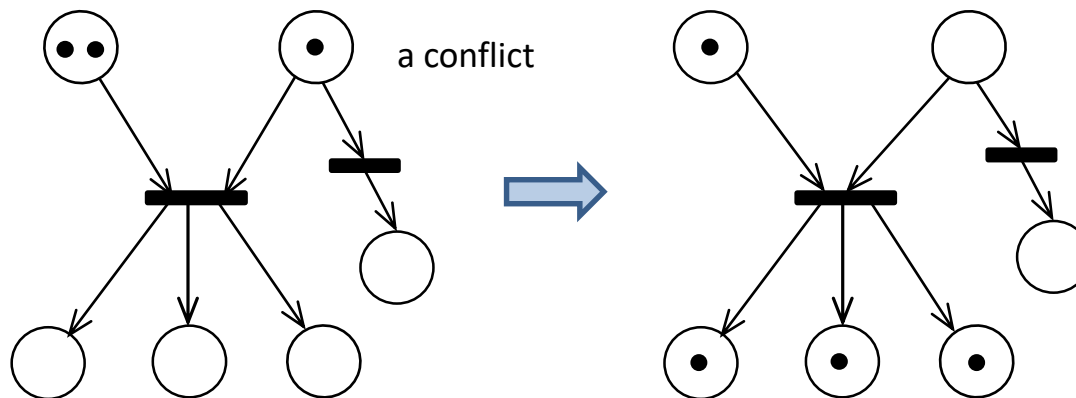
P/T-system  $\Sigma = (\mathcal{N}, M_0), \quad M_0: P \rightarrow \mathbb{N}$  - initial marking

Notation:  $\bullet t = \{p \mid (p, t) \in F\}, \quad t \bullet = \{p \mid (t, p) \in F\}$ , similarly  $\bullet p, p \bullet$

The current state is a marking  $M: P \rightarrow \mathbb{N}$ , where  $\mathbb{N} = \{0, 1, 2, \dots\}$   
(i.e. a number of tokens  $\bullet$  at each place)

Transition  $t$  is enabled in  $M$  if  $\bullet t \leq M$ , where we treat  $\bullet t$  as function  $P \rightarrow \{0, 1\}$

Transition acts as a state transformer:  $M' = M - \bullet t + t \bullet$



Each IN arc captures a token, each OUT arc deposits a token

# Petri Net, P/T-net

P/T-net is a bipartite directed graph of Places  and Transitions 

P/T-net  $\mathcal{N} = (P, T, F), \quad F \subseteq (P \times T \cup T \times P), \quad P \cap T \neq \emptyset$

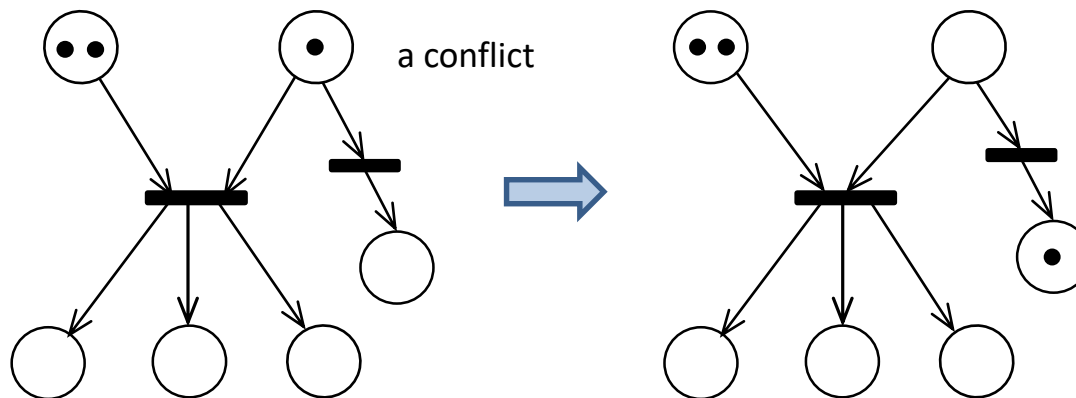
P/T-system  $\Sigma = (\mathcal{N}, M_0), \quad M_0: P \rightarrow \mathbb{N}$  - initial marking

Notation:  $\bullet t = \{p \mid (p, t) \in F\}, \quad t \bullet = \{p \mid (t, p) \in F\}$ , similarly  $\bullet p, p \bullet$

The current state is a marking  $M: P \rightarrow \mathbb{N}$ , where  $\mathbb{N} = \{0, 1, 2, \dots\}$   
(i.e. a number of tokens  $\bullet$  at each place)

Transition  $t$  is enabled in  $M$  if  $\bullet t \leq M$ , where we treat  $\bullet t$  as function  $P \rightarrow \{0, 1\}$

Transition acts as a state transformer:  $M' = M - \bullet t + t \bullet$



Each IN arc captures a token, each OUT arc deposits a token

# Petri Net, P/T-net

P/T-net is a bipartite directed graph of Places  and Transitions 

P/T-net  $\mathcal{N} = (P, T, F), \quad F \subseteq (P \times T \cup T \times P), \quad P \cap T \neq \emptyset$

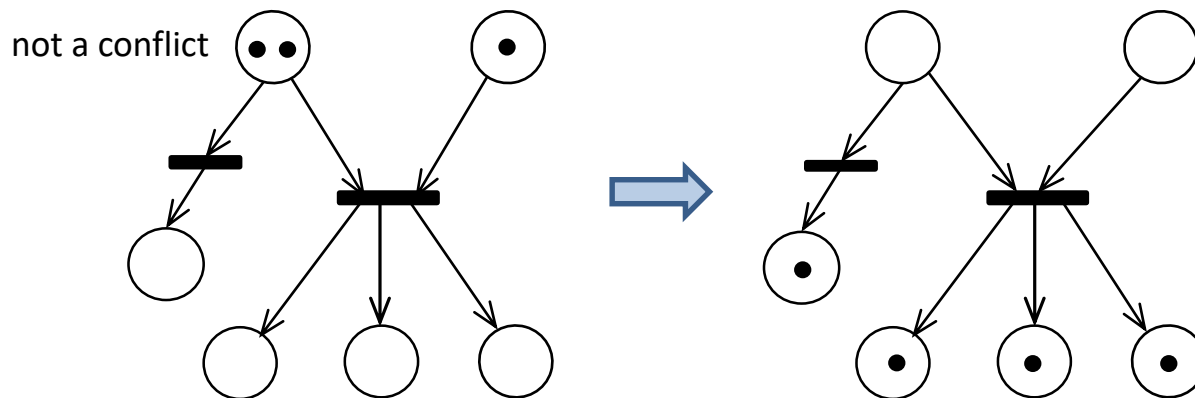
P/T-system  $\Sigma = (\mathcal{N}, M_0), \quad M_0: P \rightarrow \mathbb{N}$  - initial marking

Notation:  $\bullet t = \{p \mid (p, t) \in F\}, \quad t \bullet = \{p \mid (t, p) \in F\}$ , similarly  $\bullet p, p \bullet$

The current state is a marking  $M: P \rightarrow \mathbb{N}$ , where  $\mathbb{N} = \{0, 1, 2, \dots\}$   
(i.e. a number of tokens  $\bullet$  at each place)

Transition  $t$  is enabled in  $M$  if  $\bullet t \leq M$ , where we treat  $\bullet t$  as function  $P \rightarrow \{0, 1\}$

Transition acts as a state transformer:  $M' = M - \bullet t + t \bullet$



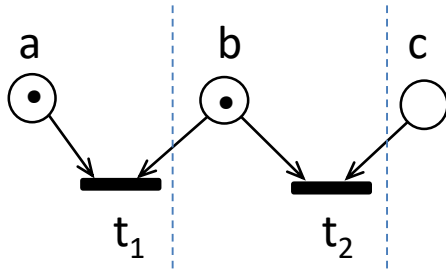
Each IN arc captures a token, each OUT arc deposits a token

# The problem of distributedness

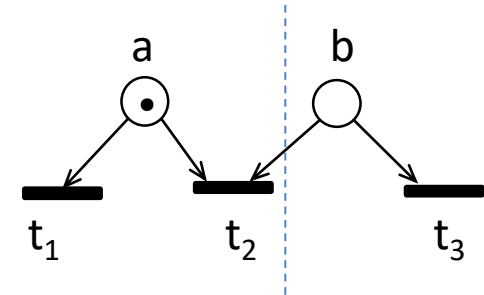
- All input places of a single transition must be captured simultaneously
- A regular (normal, good) distribution is a placement function  $f:P \rightarrow \mathbb{N}$ , that maps input places of a single transition to the same location:  $p \bullet \cap q \bullet \neq \emptyset \Rightarrow f(p)=f(q)$
- Then the mutual exclusion for synchronous capture can be provided easily, though it is too restrictive
- Consider more refined distribution with each place in a separate location
- There are several sophisticated protocols [Taubner,88], [Thomas,91] with rollbacks
- We seek for a simpler protocol without rollbacks

# Simple protocol, directed nets

Input places should be captured by each transition independently one by one in an order



Let  $t_2$  capture  $b$  first. Then  $t_1$  will be disabled with “artificial” deadlock implied (“artificial” means there was no deadlock originally)  
Here a good solution is to capture first  $a$  by  $t_1$  and  $c$  by  $t_2$



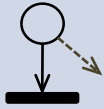
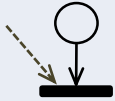
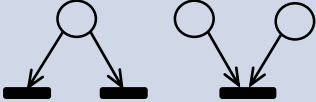
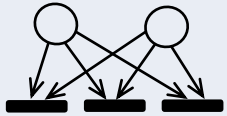
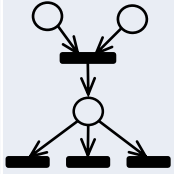
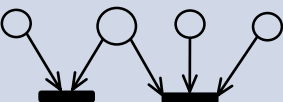
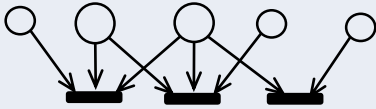
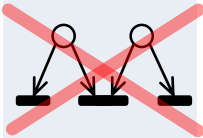
This net is not directed. Each predefined order for  $t_2$  may cause a deadlock

This net is Simple (each transition has at most one shared place).  
It is *possible to capture places asynchronously in a predefined order* (shared = last) *without a risk of artificial deadlock in any marking*

Let's call such nets *directed*. What are they structurally?

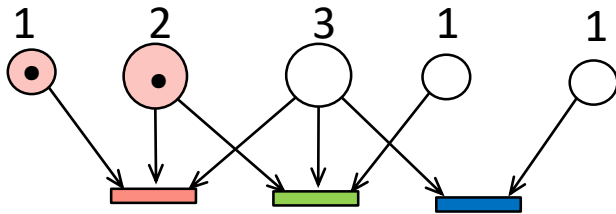


# Relevant Classes of Petri Nets

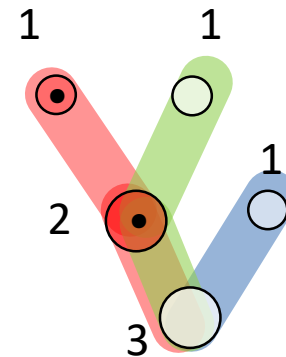
Name	Sample	Definition	Note
Join-Free		$ \bullet t =1$	
Choice Free		$ p\bullet =1$	
Free Choice (FC)		$\bullet t_1 \cap \bullet t_2 \neq \emptyset, t_1 \neq t_2$ $\Rightarrow  \bullet t_1  =  \bullet t_2  = 1$	
Extended Free Choice (EFC) Equal Conflict		$\bullet t_1 \cap \bullet t_2 \neq \emptyset \Rightarrow \bullet t_1 = \bullet t_2$ $p_1 \bullet \cap p_2 \bullet \neq \emptyset \Rightarrow p_1 \bullet = p_2 \bullet$	
Simple / Правильные		$p_1 \bullet \cap p_2 \bullet \neq \emptyset, p_1 \neq p_2$ $\Rightarrow  p_1 \bullet  = 1 \vee  p_2 \bullet  = 1$	
Asymmetric choice (AC) Extended Simple (ES)		$p_1 \bullet \cap p_2 \bullet \neq \emptyset \Rightarrow p_1 \bullet \subseteq p_2 \bullet \vee p_2 \bullet \subseteq p_1 \bullet$	

# Example

Consider the following AC net, which is not Simple.  
Shared places are larger circles



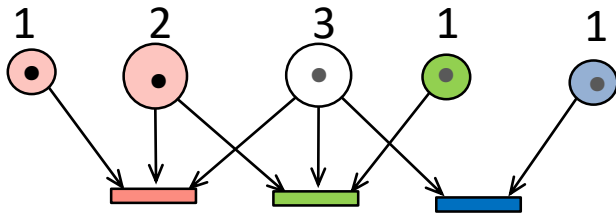
Digits indicate order of places, i.e.  
number of output transitions. Let them  
be the capture ordering as well.  
The “red” transition will capture the  
two tokens.  
Now let tokens to other three places  
come



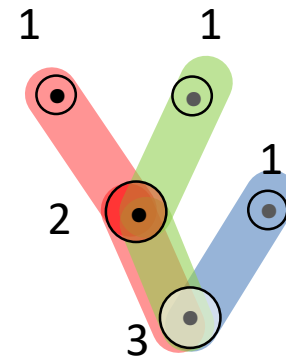
The same net with transitions  
shown as colored bands

# Example

Consider the following Asymmetric Choice net, which is not Simple. Shared places are shown as larger circles



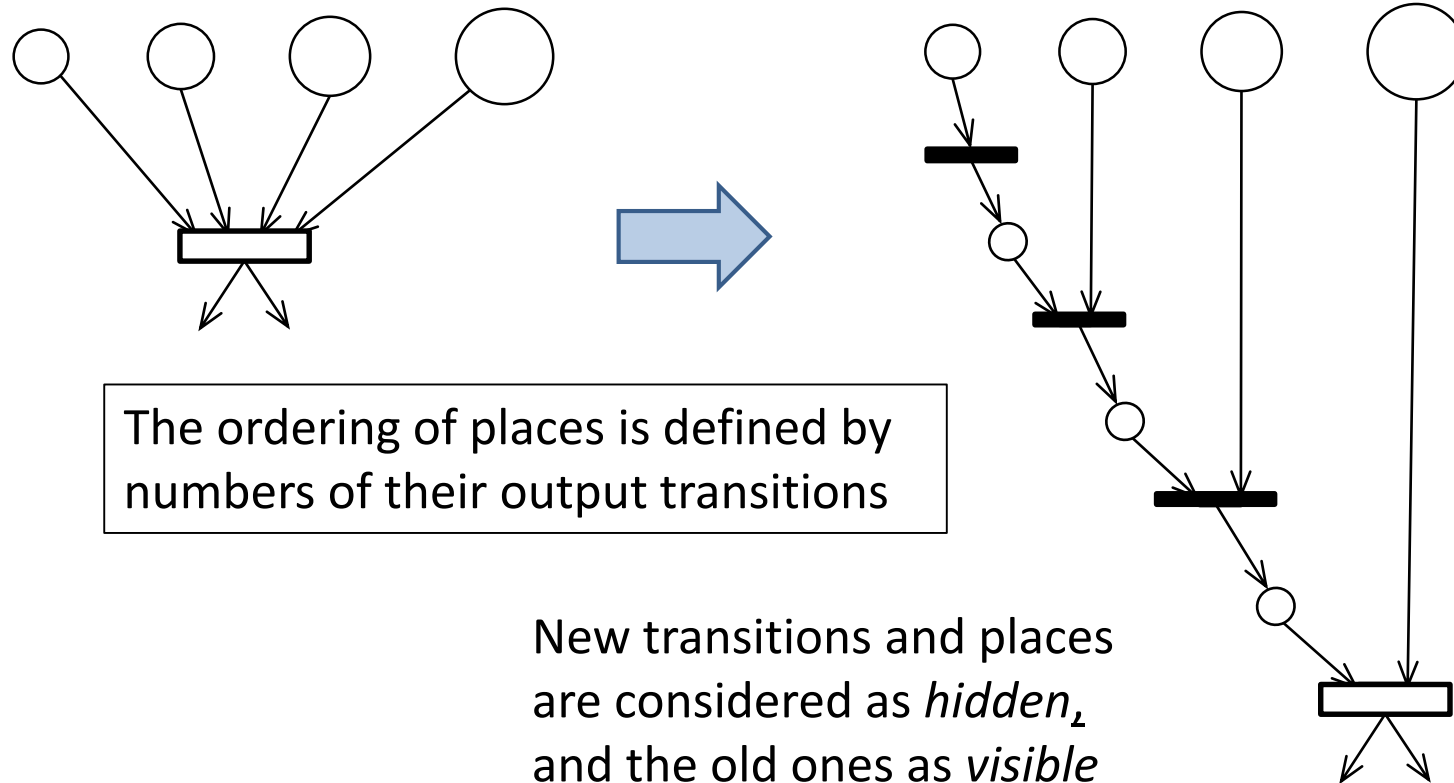
The “green” transition can’t capture its 2-place as it is captured by “red”. So it won’t be enabled unless the “red” transition fire and new tokens come. The 3-place can now be captured by either “red” or “blue” transition. In any case one of them will be enabled and hence artificial deadlock will not occur.



The same net with transitions shown as colored bands

# The simulation of Directed net behavior by Simple net

Each transition with several places is transformed as follows



The resulting net is Simple as only old places may remain shared

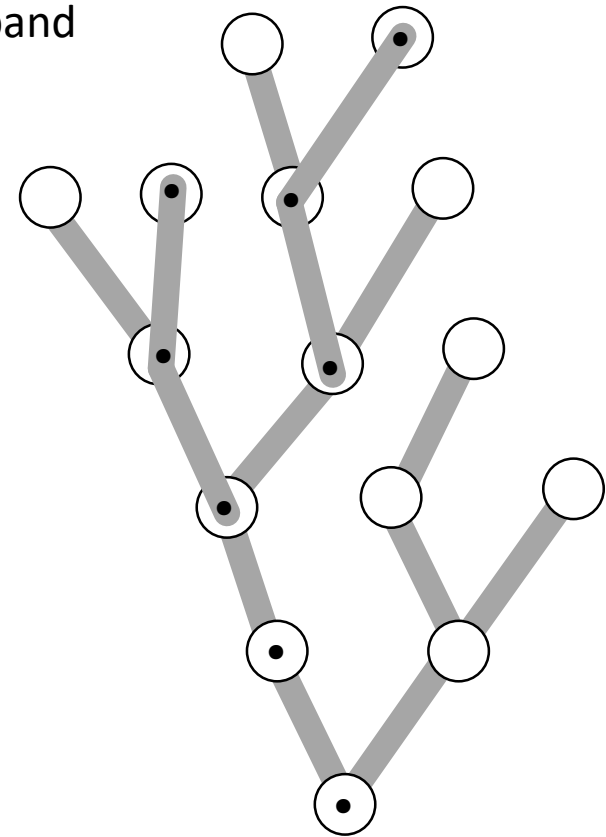
# The general tree view of Asymmetric Choice net CC (connected component)

- Each transition is a path from the root to a place
- Each leaf forms a unique transition
- Each transition captures its tokens from top to root
- Captured tokens are shown as continuous gray band

**Lemma.**

*If enabled transition exist in the original net then either root is captured or at least one new token can be captured.*

**Proof.** Consider enabled path and the lowest captured place on it (if there are none, the path can capture its first token). If it is not the root then the path that captured it can capture the next token



**Corollary.** *No artificial deadlock can occur.*

# Does semantics preserve?

- Usual concept of simulation does not hold
- The problem is that hidden transitions are irreversible and may disable some of the conflicting enabled transitions in original net
- The new *weak simulation* concept is proposed
- The idea is to define the *base set* B of markings of simulating net which
  - a. has the *home* property and
  - b. allows all of the enabled transitions in the corresponding markings of the original net
- Normal simulation (which does not hold in our case) can be obtained by setting B=all markings
- The two problems are left:
  1. To justify the concept of weak simulation
  2. To prove that with the above transform we indeed get the net that weakly simulates the original one (it is easy in case of live net, but hard otherwise)

Spasibo!

?