Branching processes of conservative nested Petri nets

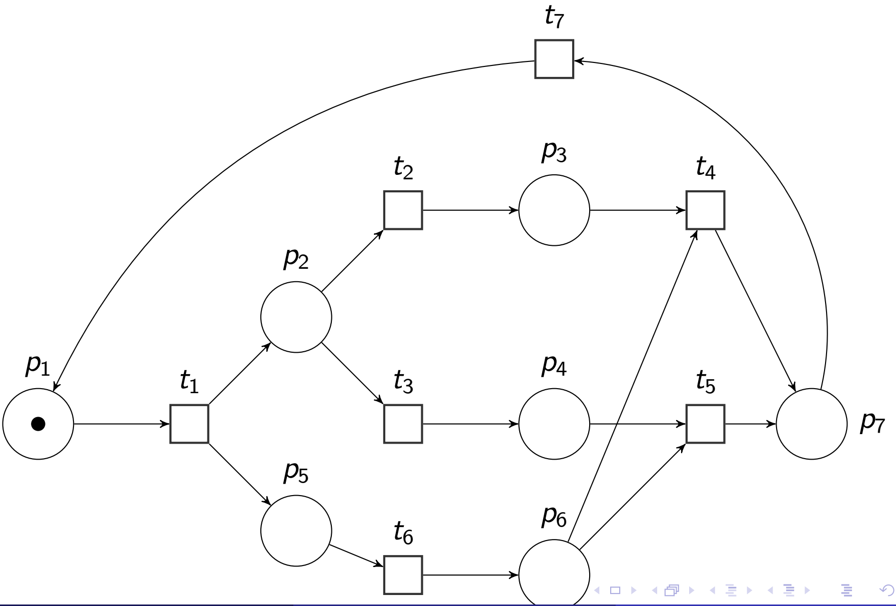
Daniil Frumin, Irina Lomazova

International Laboratory of Process-Aware Information Systems
National Research University Higher School of Economics

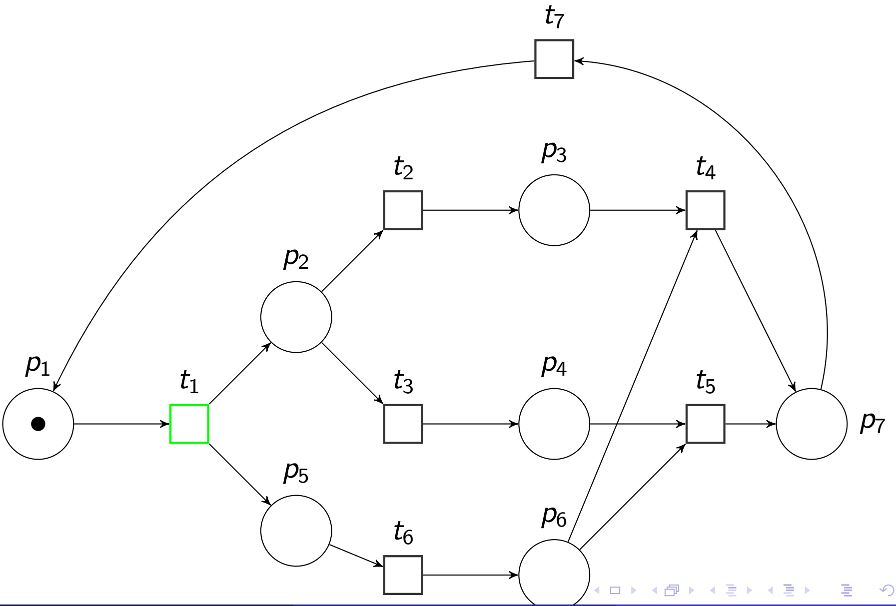
July 17, 2014

Talk overview

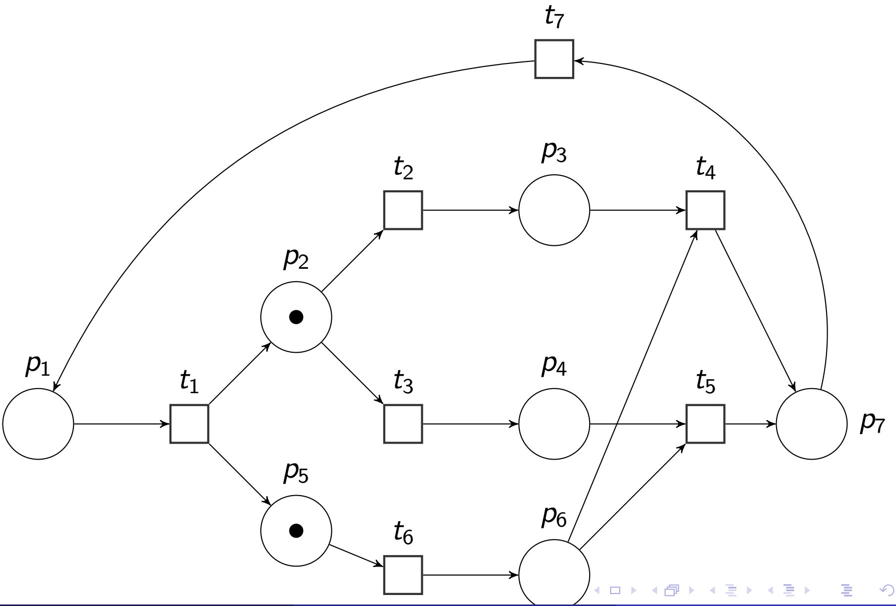
Petri nets



Petri nets



Petri nets



Petri nets (definition)

$N = (P, T, F, M_0)$

- P and T are disjoint sets of *places* and *transitions*;
- $F \subseteq (P \times T) \cup (T \times P)$ is a *flow relation*;
- $M_0 \subseteq P$ is an *initial marking* of N.

Pre- and post-set functions are defined for each $x \in T$:

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

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

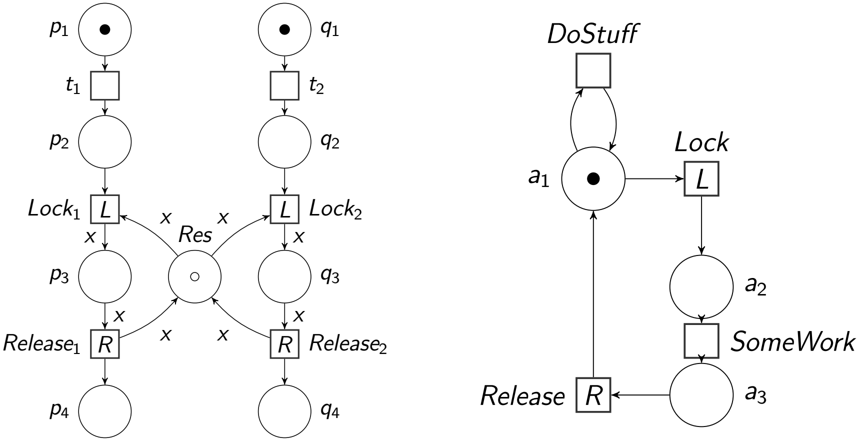
Petri nets (behavior)

A transition t in the Petri net $N = (P, T, F, M_0)$ is *active* under a marking M iff ${}^\bullet t \subseteq M$.
An active transition may *fire*, leading to a marking $M' = M - {}^\bullet t + t^\bullet$, denoted as $M \xrightarrow{t} M'$.
A marking M is *reachable* (from the initial marking M_0) iff there exists a sequence of firings $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \rightarrow \cdots \rightarrow M$ leading to it.

Nested Petri nets

- The "flat" structure of regular Petri nets can be inconvenient, when modelling systems with multiple autonomous agents.
- Nested Petri net (NP-net) is an extension of classical Petri nets used for modelling dynamic multi-agent systems.
- In NP-net tokens can be Petri nets themselves ("nets-within-nets" approach).

Nested Petri nets



NP-net = system net + element nets

An instance of the element nets are called net tokens.

Nested Petri nets: formally

An NP-net NP is a tuple $(SN, (EN_1, \ldots, EN_k), \upsilon, \lambda, W)$, where

- $SN = (P_{SN}, T_{SN}, F_{SN})$ is a Petri net called a *system net*.
- For each $i = \overline{1, k}$, $EN_i = (P_{EN_i}, T_{EN_i}, F_{EN_i})$ is a Petri net called an *element net*, where all the sets of places and transitions are disjoint; each element net is assigned a type from $Type$.
- $\upsilon : P_{SN} \to Type \cup \{\bullet\}$ is a type assignment function
- $\lambda : T_{NP} \to Lab$ is a partial labeling function, where $T_{NP} = T_{SN} \cup T_{EN_1} \cup \cdots \cup T_{EN_k}$;
- $W : F_{SN} \to Var \cup \{\bullet\}$ is an *arc labeling function* s.t. for an arc r adjacent to a place p the type of $W(r)$ coincides with the type of p.

A marked element net is called a *net token*.

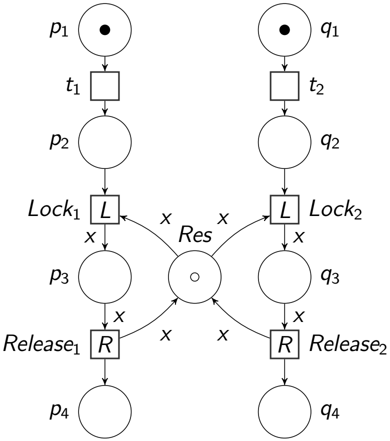
NPN markings

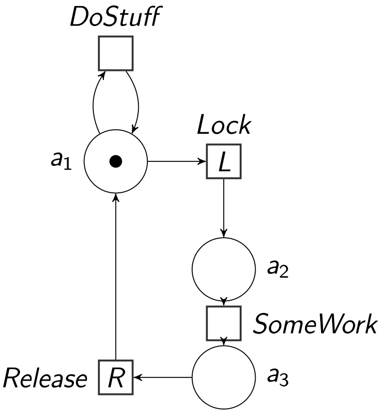
A marking of an NP-net maps each place of the system net to a multiset of regular or net tokens. Marking should respect the typing.

NPN markings

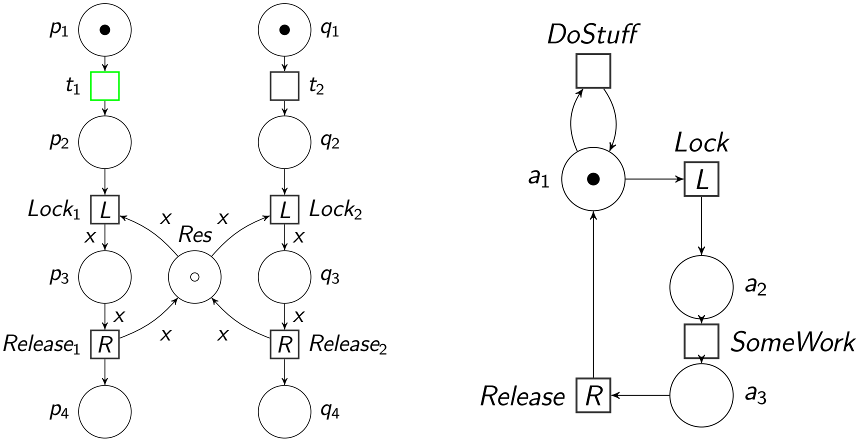
A marking of an NP-net maps each place of the system net to a multiset of regular or net tokens. Marking should respect the typing.
$M : P_{SN} \to \mathcal{M}(A \cup \{\bullet\})$
where $A = \{(EN_i, \mu_i) \mid \mu_i \text{ is a marking of } EN_i\}$

NP-net behaviour: system-autonomous step

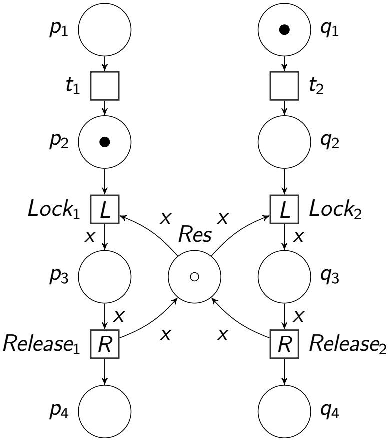


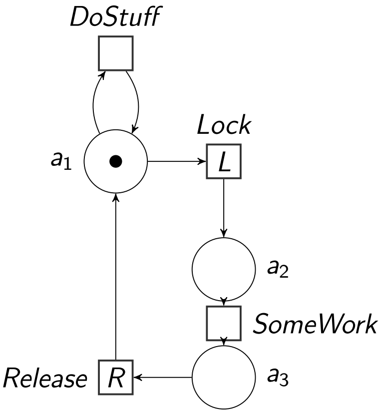


NP-net behaviour: system-autonomous step

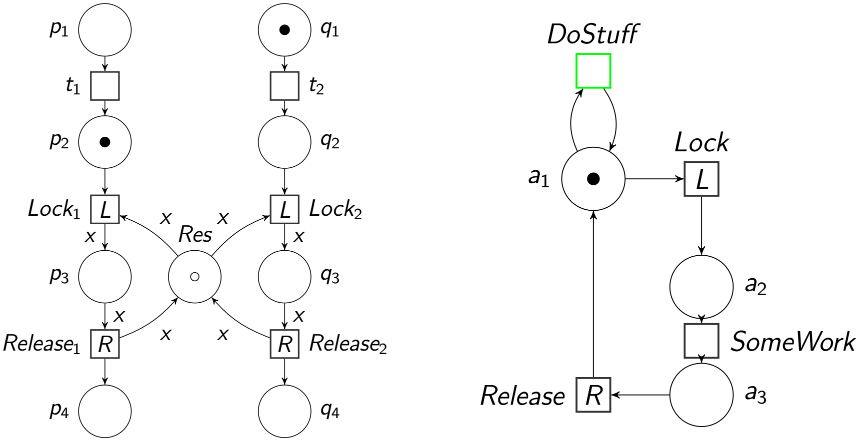


NP-net behaviour: system-autonomous step

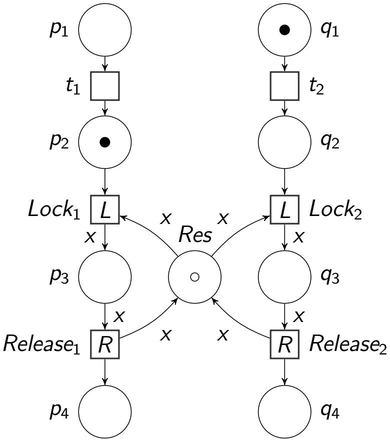


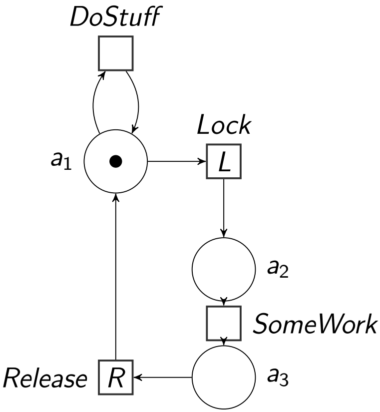


NP-net behaviour: element-autonomous step

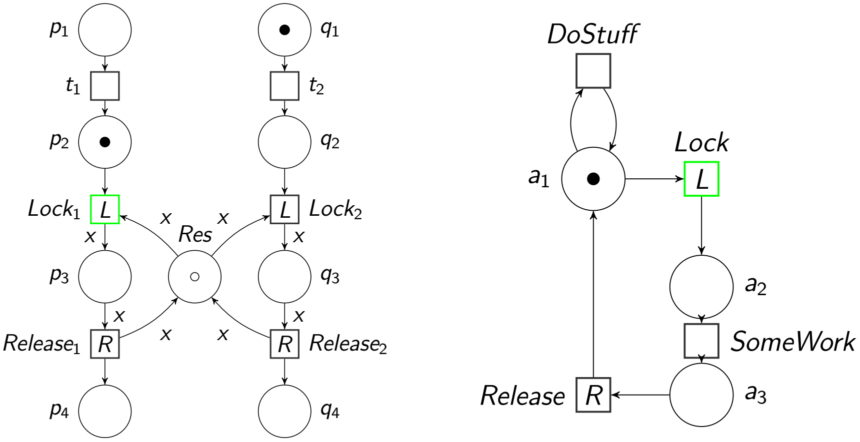


NP-net behaviour: element-autonomous step

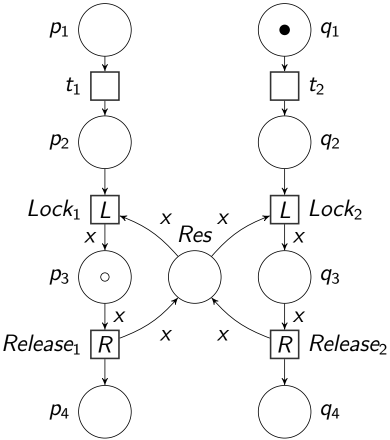


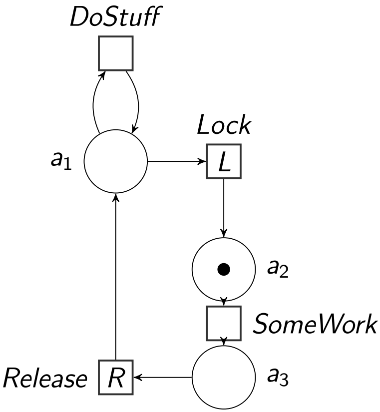


NP-net behaviour: synchronization step



NP-net behaviour: synchronization step

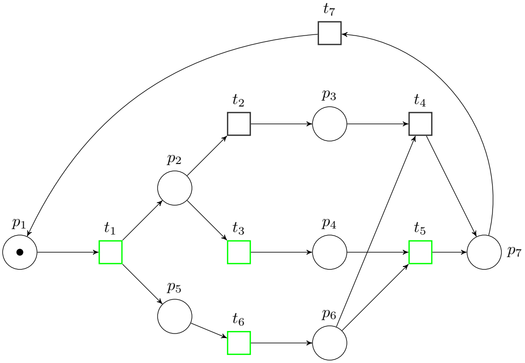




Talk overview

True concurrency and Petri nets

- Sequential execution: t_1, t_3, t_6, t_5 and t_1, t_6, t_3, t_5
- There may be several sequential executions corresponding to one set of transitions.



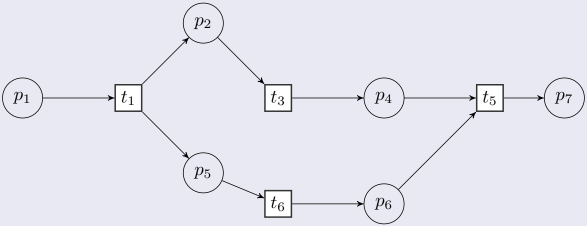
True concurrency

True concurrency vs interleavings

- Non-true concurrency semantics for process algebras:
  $a \parallel b \simeq a.b + b.a$
- True concurrency semantics distinguish between parallel composition and non-deterministic choice between interleavings.

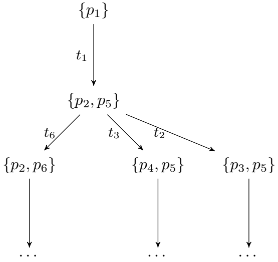
Non-sequential processes

Non-sequential processes captures *concurrent* runs of the net.

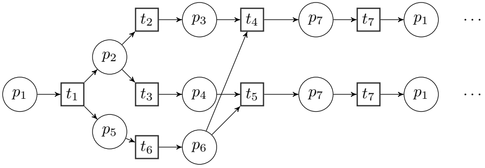


Unfoldings and computational trees

All the sequential executions of the net can be bundled in a *computational tree*.



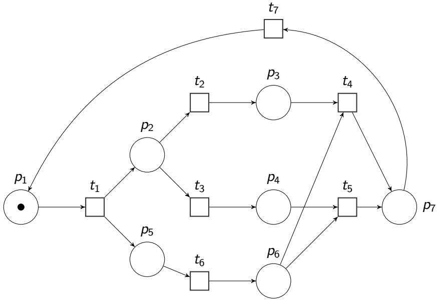
All the non-sequential processes of the net can be bundled in an *unfolding*.



Unfoldings in verification

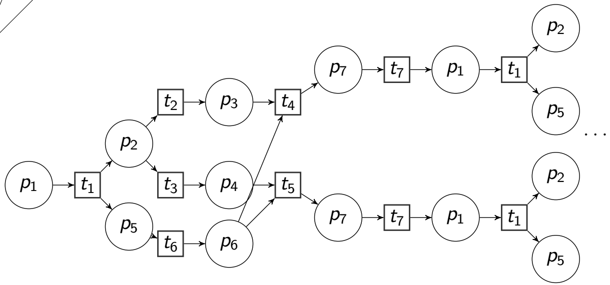
- Unfoldings of Petri nets provide true concurrency semantics to Petri nets.
- If we can "cut" the unfolding to a finite prefix, then we can use it for verification.
- The size of finite prefixes of unfoldings can be much smaller than the size of the reachability graph.

Unfoldings

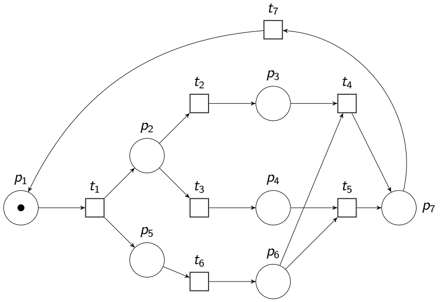


Unfoldings are represented by a special class of Petri nets - Occurrence nets.

Occurrence nets are acyclic, the flow relation induces a partial order $<$: transitive closure of F.

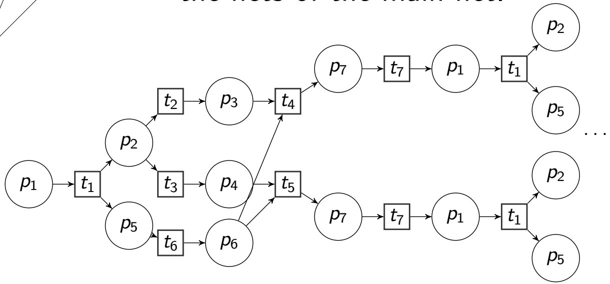


Unfoldings

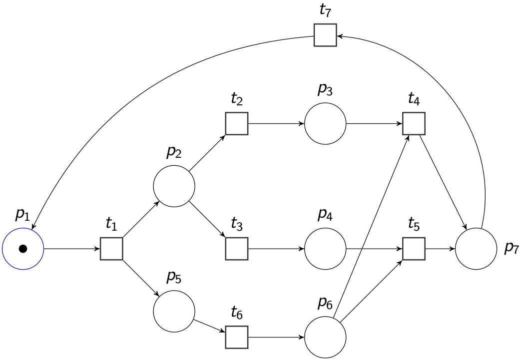


Unfoldings are defined using *branching processes* – partial branching concurrent runs of the system. The function h relates the nodes of a branching process to the nets of the main net.

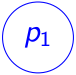
Places and transitions in a branching process are called *conditions* and *events*.



Branching processes

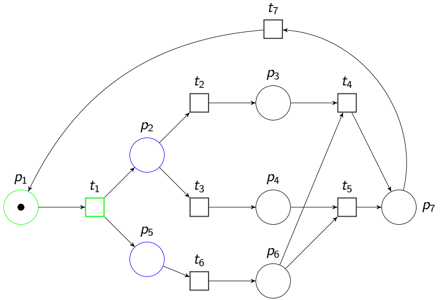


A net, consisting just of places, corresponding to the initial marking of N, is a branching process.



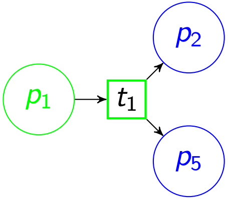
The initial marking of a branching process is the $<$-minimal set.

Branching processes

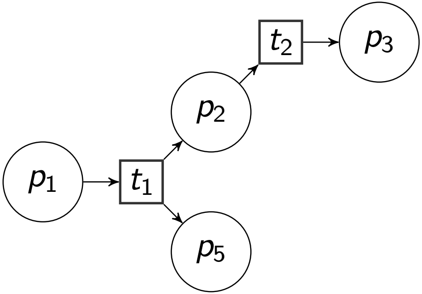


If X is a set of reachable conditions of a branching process B, and X correspond to a set $h(X)$ of places in the net N, that enable a transition t..

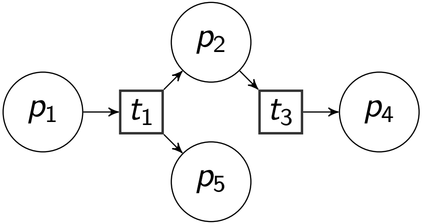
.. Then we can obtain a branching process B' by adding a new event e (which corresponds to t), and "fresh" post-conditions which correspond to $t^\bullet$. Such e is called a *possible extension* of B.

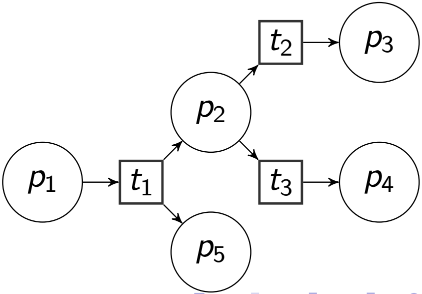


Branching processes



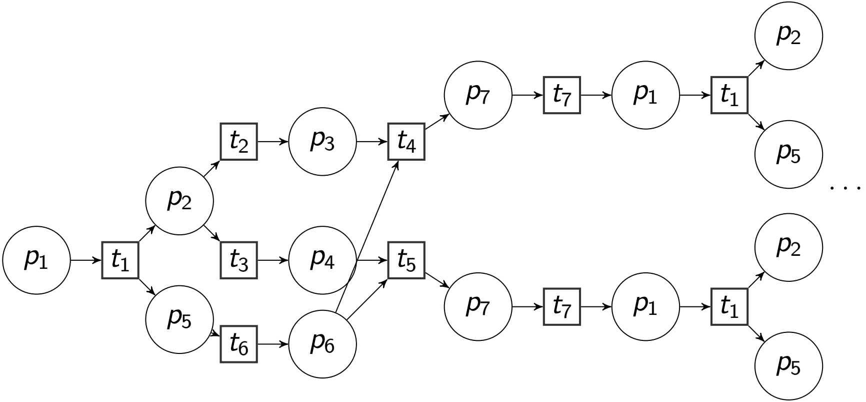
Let BB be a (finite or infinite) set of branching processes. The net $\bigcup BB$ is a branching process.





Net unfoldings

A branching process B_1 is said to be a *prefix* of a branching process B_2 (denoted as $B_1 \sqsubseteq B_2$), iff B_1 can be "included" in B_2



The maximal (w.r.t. $\sqsubseteq$) branching process is called an *unfolding*, and is denoted by $U(N)$.

Net unfoldings

Fundamental property of unfoldings

Let M be a reachable marking of N, and M_U be a reachable marking of $U(N)$, such that M_U correspond to M.

1. if there is a step $M_U \xrightarrow{t_U} M'_U$ of $U(N)$, then there is a step $M \xrightarrow{t} M'$ of N, such that $h(t_U) = t \wedge h(M'_U) = M'$;
2. if there is a step $M \xrightarrow{t} M'$ of N, then there is a step $M_U \xrightarrow{t_U} M'_U$ in $U(N)$, such that $h(t_U) = t \wedge h(M'_U) = M'$.

Nested Petri nets and unfoldings

- NP-nets are good for modeling multi-agent systems
- Multi-agent systems inherently posses a high grade of concurrency
- Unfolding NP-nets can be beneficial compared to state-space exploration

Talk overview

Conservative NP-nets

Here we deal with safe conservative NP-nets.

- A net N is called *safe* iff $\forall M \in \mathcal{RM}(N)$, $M(p) \leqslant 1$.
- A net N is called conservative iff any transition firing does not change the number of net tokens in the system net (inner markings of the net tokens can be changed).

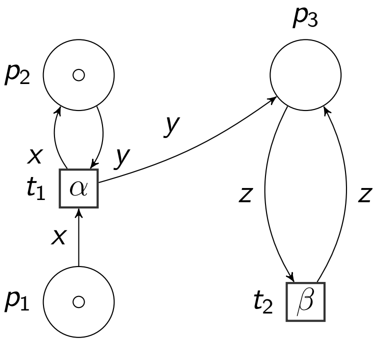
Conservative NP-nets

Here we deal with safe conservative NP-nets.

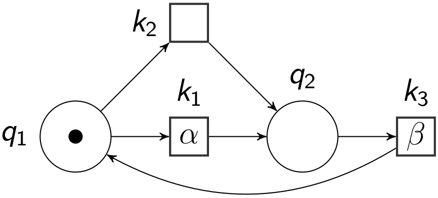
- A net N is called *safe* iff $\forall M \in \mathcal{RM}(N)$, $M(p) \leqslant 1$.
- A net N is called conservative iff any transition firing does not change the number of net tokens in the system net (inner markings of the net tokens can be changed).
- 1. For all $t \in T_{SN}$ and for all $p \in {}^{\bullet}t$, $\exists! p' \in t^{\bullet} \,.\, W(p,t) = W(t,p')$ or $W(p,t)$ *bullet*.
  2. For all $t \in T_{SN}$ and for all $p \in t^{\bullet}$, $\exists! p' \in {}^{\bullet}t \,.\, W(t,p) = W(p',t)$ or $W(t,p)$ *bullet*.
- Such place p' is said to be *adjacent to p via t*.

NP-nets unfoldings

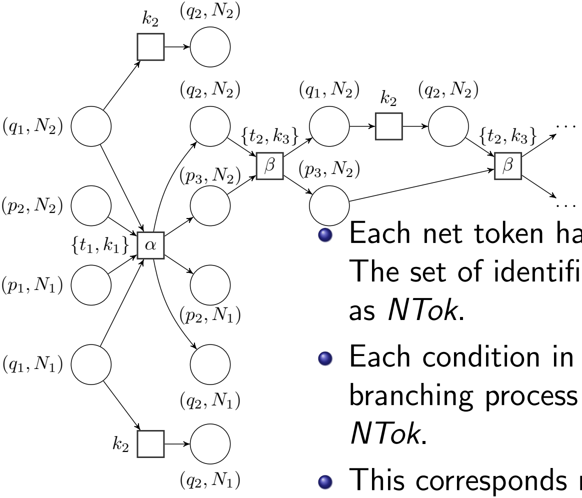
Unfoldings for NP-net are defined using branching processes, similarly to the case of classical Petri nets.



NP_2: system net

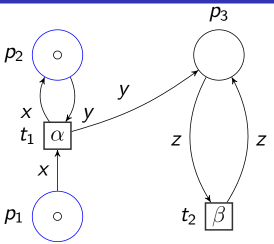


NP_2: element net

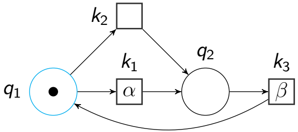


- Each net token has an ID assigned to it. The set of identified net tokens is denoted as *NTok*.
- Each condition in the element-indexed branching process is paired with an id from *NTok*.
- This corresponds nicely to the intuitive meaning of “conditions” and “events” in occurrence nets.

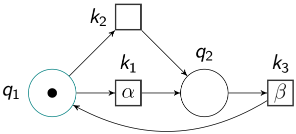
Initial branching process



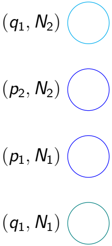
System net



Net token N_1

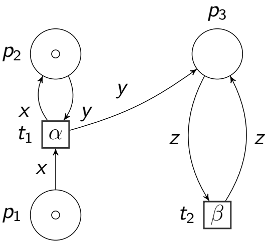


Net token N_2

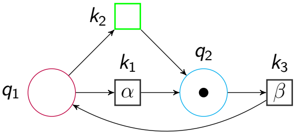


Initial b-process

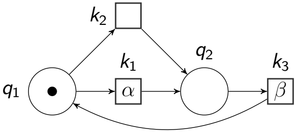
Possible extensions of branching processes



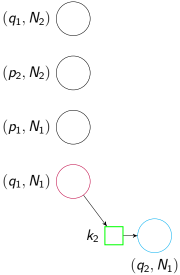
System net



Net token N_1

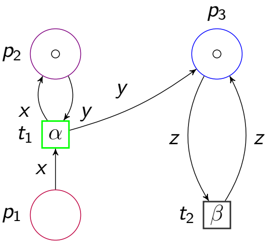


Net token N_2

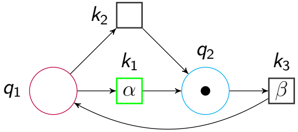


Branching process

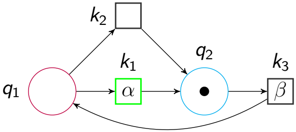
Possible extensions of branching processes



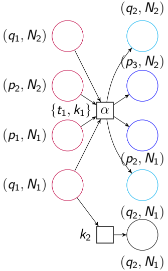
System net



Net token N_1

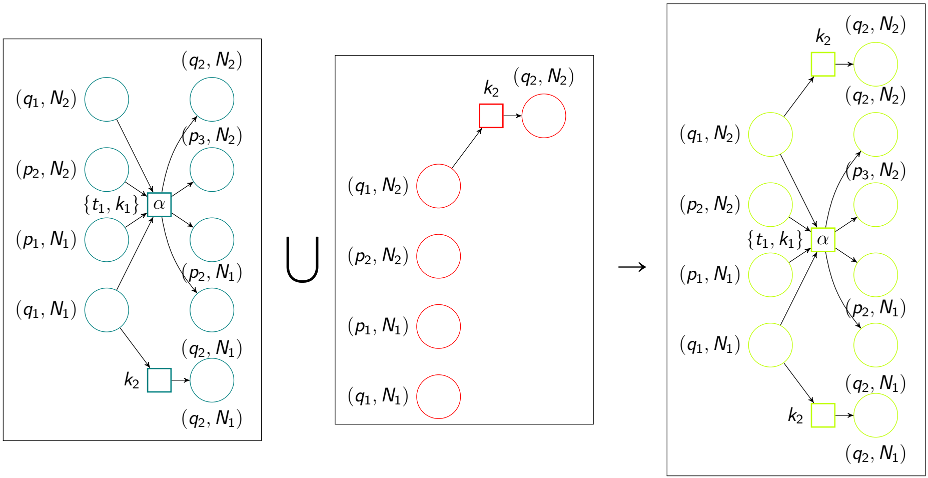


Net token N_2

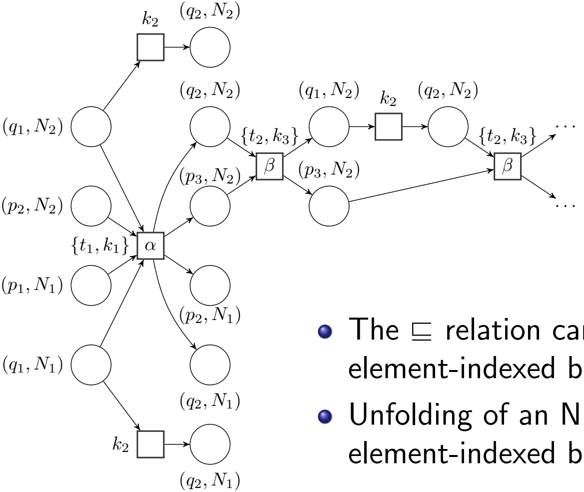


Branching process

Union of branching processes



NP-nets unfoldings



- The $\sqsubseteq$ relation can be generalized to element-indexed branching processes
- Unfolding of an NP-net NP is the maximal element-indexed branching process $U(NP)$

Properties of branching processes

Property

Every element-indexed branching process is an occurrence net.

Property

A flat P/T-net is a special case of an NP-net with the empty set of element nets and no vertical synchronization.
Let N be a P/T-net. The set of branching processes of N is isomorphic to the set of element-indexed branching processes of N, when N is considered as an NP-net.

Property

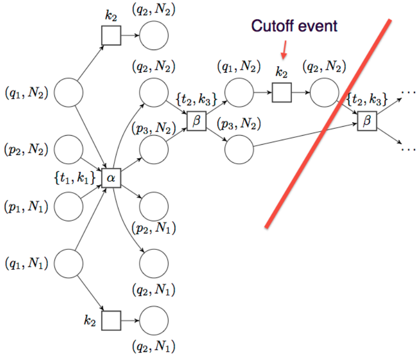
The behaviour of the unfolding is isomorphic to the behaviour of the net.

Verification with branching processes

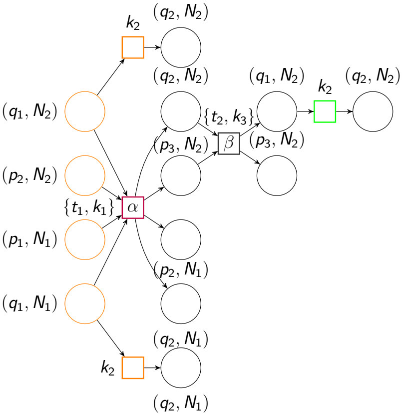
- The theory of *canonical prefixes* can be directly applied to the element-indexed branching processes.
- The existing algorithms can be applied with minor changes.

Finite prefix generation example

$$C' \approx C'' \iff Mark(C') = Mark(C'') \text{ and } C' \lhd C'' \iff |C'| < |C''|$$



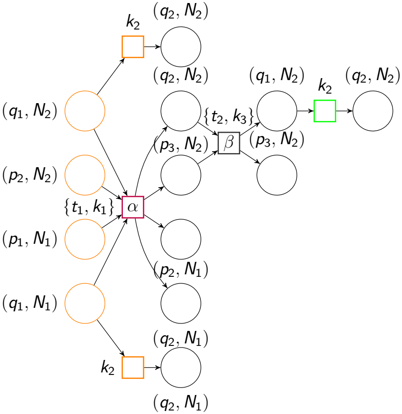
Unfolding



Finite prefix

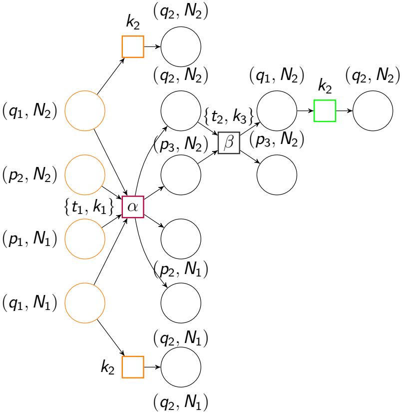
Execution problem

- Can a transition t be executed in the net?
- We just have to check if BP_c has a transition labeled by t.



Deadlock problem

- Is there a deadlock in the net?
- The net has a deadlock iff there exists a configuration in the prefix that does not contain cut-off events and the corresponding marking is a deadlock in the prefix.



Talk overview

Future work

- Extending the technique to (a bigger) NP-net classes.
- Trying a more algebraic and compositional approach.

Thank you for your attention!

PAIS lab

http://pais.hse.ru/en/