Verification of systems communicating via unbounded channels

R. Gascon⋆, É. Madelaine⋆ & V. Maisonneuve⋆⋆

⋆ INRIA - Projet OASIS & ⋆⋆ ENS Cachan

SAFA Workshop - December 3rd, 2008

VERCORS in a nutshell

- Platform for specification of distributed applications.
- Based on the semantics features of the ProActive library.

  `http://www-sop.inria.fr/oasis/ProActive/`

- Generation of intermediate finite model.
- Various tools can then operate on these models:
  static analysis, model checking, code generation...
- The aim is to integrate the platform in a development environment, used by non-specialists.

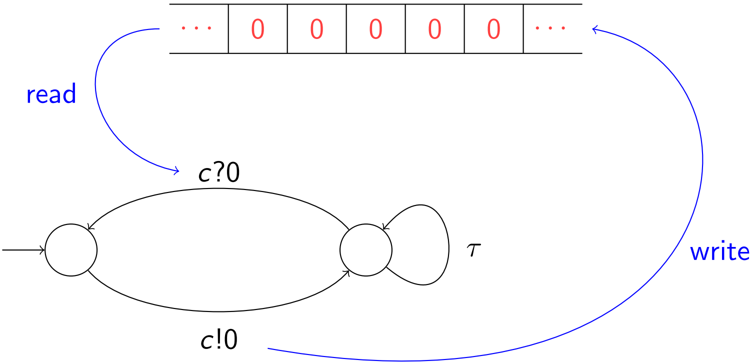
Formal verification of pNets

- Basically, pNets are made of LTSs synchronized by mean of transducer (synchronization vector).
- Verifying pNets remains to verifies systems:
  - manipulating unbounded data,
  - having a parameterized topology,
  - using unbounded communication queues.
- Numerous sources of infinity
  $\Leftrightarrow$ numerous complications for formal verification.
- Current platform uses only finite-sate based model-checkers.
- We want to apply infinite state model-checking techniques.

Outline

Communicating finite state machines

Basically a finite state machine augmented with a set of queues.



Communicating finite state machines

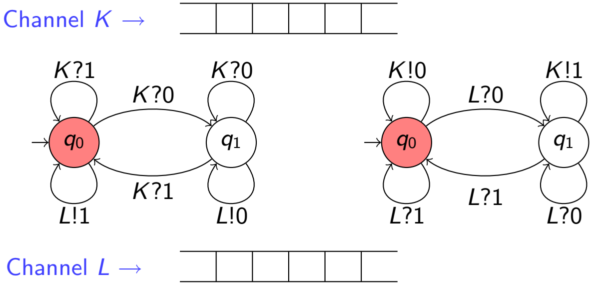
Formally, a communicating finite state machines (CFSM) is a tuple

$$\mathcal{M} = (Q, q_0, C, \Sigma, A, \delta) \quad \text{such that}$$

- $Q =$ is a finite set of states,
- $q_0 \in Q$ is the initial state,
- C is a set of communicating channels/queues,
- Σ is the alphabet of messages,
- A is a finite set of internal actions,
- $\delta \subset Q \times ((C \times \{?, !\} \times \Sigma) \cup A) \times Q$ is the transition relation.

Short Example

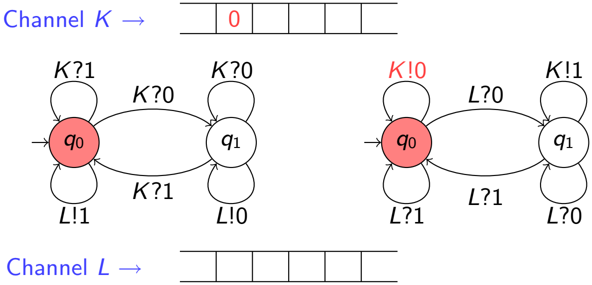
- Execution: Sequence respecting the transition relation.



- $\langle q_0, q_0, \varepsilon, \varepsilon \rangle$

Short Example

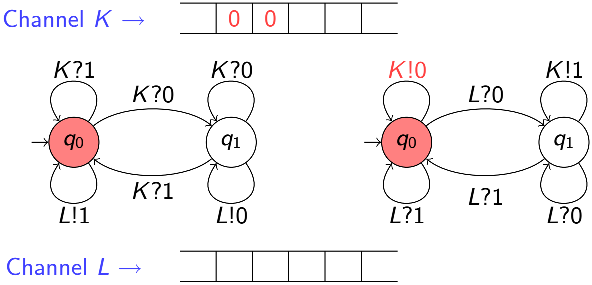
- Execution: Sequence respecting the transition relation.



- $\langle q_0, q_0, \varepsilon, \varepsilon \rangle \xrightarrow{K!0} \langle q_0, q_0, 0, \varepsilon \rangle$

Short Example

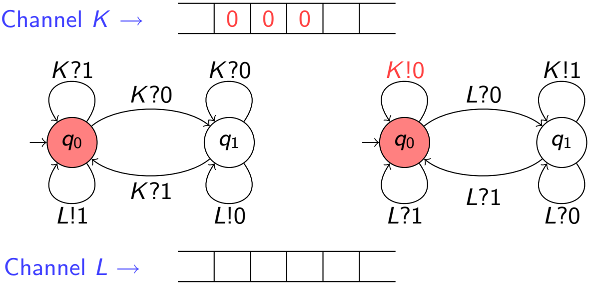
- Execution: Sequence respecting the transition relation.



- $\langle q_0, q_0, \varepsilon, \varepsilon\rangle \xrightarrow{K!0} \langle q_0, q_0, 0, \varepsilon\rangle \xrightarrow{K!0} \cdots \xrightarrow{K!0}$

Short Example

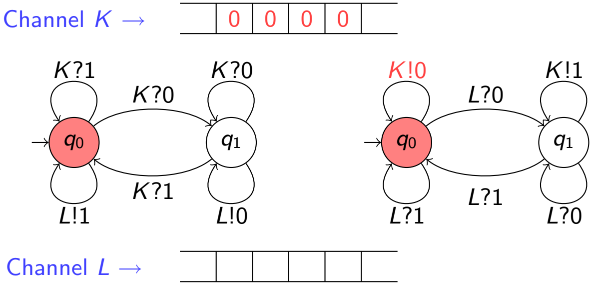
- Execution: Sequence respecting the transition relation.



- $\langle q_0, q_0, \varepsilon, \varepsilon \rangle \xrightarrow{K!0} \langle q_0, q_0, 0, \varepsilon \rangle \xrightarrow{K!0} \cdots \xrightarrow{K!0}$

Short Example

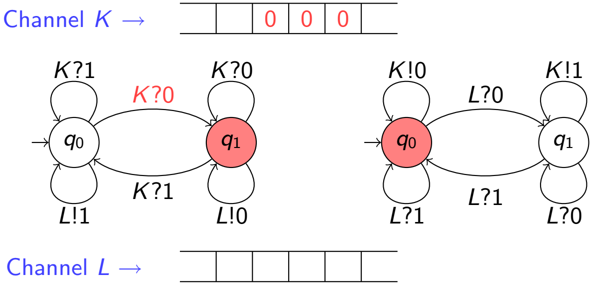
- Execution: Sequence respecting the transition relation.



- $\langle q_0, q_0, \varepsilon, \varepsilon \rangle \xrightarrow{K!0} \langle q_0, q_0, 0, \varepsilon \rangle \xrightarrow{K!0} \cdots \xrightarrow{K!0} \langle q_0, q_0, 0000, \varepsilon \rangle$

Short Example

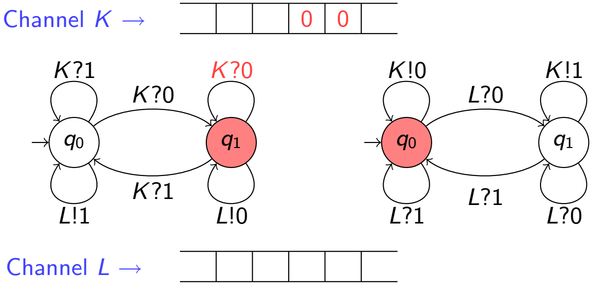
- Execution: Sequence respecting the transition relation.



- $\langle q_0, q_0, \varepsilon, \varepsilon\rangle \xrightarrow{K!0} \langle q_0, q_0, 0, \varepsilon\rangle \xrightarrow{K!0} \cdots \xrightarrow{K!0} \langle q_0, q_0, 0000, \varepsilon\rangle \xrightarrow{K?0} \langle q_1, q_0, 000, \varepsilon\rangle$

Short Example

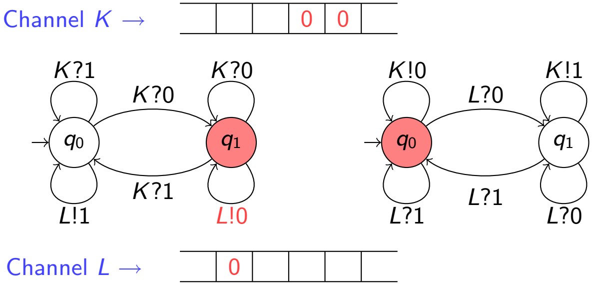
- Execution: Sequence respecting the transition relation.



- $\langle q_0, q_0, \varepsilon, \varepsilon\rangle \xrightarrow{K!0} \langle q_0, q_0, 0, \varepsilon\rangle \xrightarrow{K!0} \cdots \xrightarrow{K!0} \langle q_0, q_0, 0000, \varepsilon\rangle \xrightarrow{K?0}$ $\langle q_1, q_0, 000, \varepsilon\rangle \xrightarrow{K?0} \langle q_1, q_0, 00, \varepsilon\rangle$

Short Example

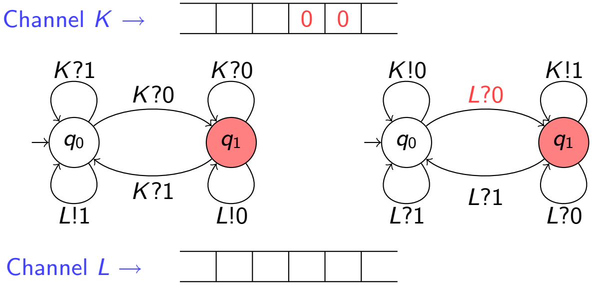
- Execution: Sequence respecting the transition relation.



- $\langle q_0, q_0, \varepsilon, \varepsilon\rangle \xrightarrow{K!0} \langle q_0, q_0, 0, \varepsilon\rangle \xrightarrow{K!0} \cdots \xrightarrow{K!0} \langle q_0, q_0, 0000, \varepsilon\rangle \xrightarrow{K?0}$ $\langle q_1, q_0, 000, \varepsilon\rangle \xrightarrow{K?0} \langle q_1, q_0, 00, \varepsilon\rangle \xrightarrow{L!0} \langle q_1, q_0, 00, 0\rangle$

Short Example

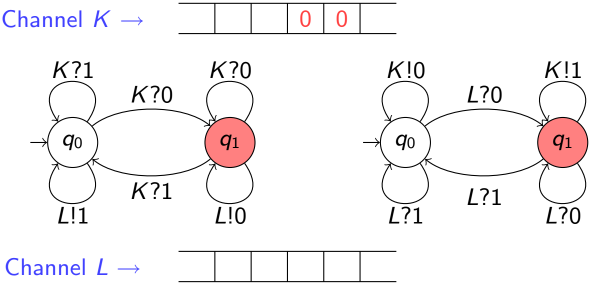
- Execution: Sequence respecting the transition relation.



- $\langle q_0, q_0, \varepsilon, \varepsilon\rangle \xrightarrow{K!0} \langle q_0, q_0, 0, \varepsilon\rangle \xrightarrow{K!0} \cdots \xrightarrow{K!0} \langle q_0, q_0, 0000, \varepsilon\rangle \xrightarrow{K?0}$ $\langle q_1, q_0, 000, \varepsilon\rangle \xrightarrow{K?0} \langle q_1, q_0, 00, \varepsilon\rangle \xrightarrow{L!0} \langle q_1, q_0, 00, 0\rangle \xrightarrow{L!0}$ $\langle q_1, q_1, 00, \varepsilon\rangle$

Short Example

- Execution: Sequence respecting the transition relation.



- $\langle q_0, q_0, \varepsilon, \varepsilon\rangle \xrightarrow{K!0} \langle q_0, q_0, 0, \varepsilon\rangle \xrightarrow{K!0} \cdots \xrightarrow{K!0} \langle q_0, q_0, 0000, \varepsilon\rangle \xrightarrow{K?0}$
  $\langle q_1, q_0, 000, \varepsilon\rangle \xrightarrow{K?0} \langle q_1, q_0, 00, \varepsilon\rangle \xrightarrow{L!0} \langle q_1, q_0, 00, 0\rangle \xrightarrow{L!0}$
  $\langle q_1, q_1, 00, \varepsilon\rangle \rightarrow \cdots$

Operational Semantics

- We consider unbounded FIFO queues.
- Consider a set of CFSM sharing a set of queues $\{K, L\}$.
- Configuration: $\langle q_1, q_2, w_K, w_L \rangle$ (for a pair of CFSM)

  Global state + Queue contents
- Operations:
  - Send (non-blocking).

    if $\langle q_1, K!a, q_1' \rangle \in \delta_1$ then

$$\langle q_1, q_2, w_K, w_L \rangle \xrightarrow{K!a} \langle q_1', q_2, w_K \cdot a, w_L \rangle$$

  - Receive (blocking).
  - Internal Action.

Operational Semantics

- We consider unbounded FIFO queues.
- Consider a set of CFSM sharing a set of queues $\{K, L\}$.
- Configuration: $\langle q_1, q_2, w_K, w_L \rangle$ (for a pair of CFSM)

  Global state + Queue contents
- Operations:
  - Send (non-blocking).
  - Receive (blocking).

    if $\langle q_1, K?a, q_1' \rangle \in \delta_1$ then

$$\langle q_1, q_2, a \cdot w_K, w_L \rangle \xrightarrow{K!a} \langle q_1', q_2, w_K, w_L \rangle$$

  - Internal Action.

Operational Semantics

- We consider unbounded FIFO queues.
- Consider a set of CFSM sharing a set of queues $\{K, L\}$.
- Configuration: $\langle q_1, q_2, w_K, w_L \rangle$ (for a pair of CFSM)

  Global state + Queue contents
- Operations:
  - Send (non-blocking).
  - Receive (blocking).
  - Internal Action.

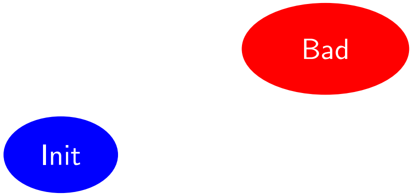
    if $\langle q_1, \tau, q_1' \rangle \in \delta_1$ with $\tau \in A$ then

$$\langle q_1, q_2, w_K, w_L \rangle \xrightarrow{\tau} \langle q_1', q_2, w_K, w_L \rangle$$

Outline

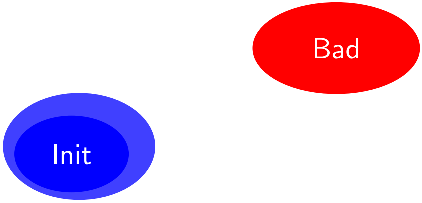
Reachability Problem

We consider the following problem:



Reachability Problem

We consider the following problem:

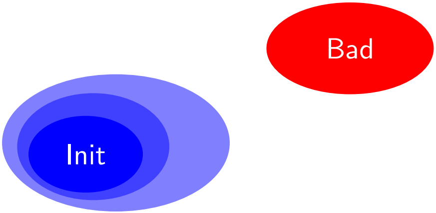


We note:

- $\mathrm{Post}(X) = \{x \mid \exists x' \in X \text{ s.t. } x \rightarrow x'\}$.

Reachability Problem

We consider the following problem:

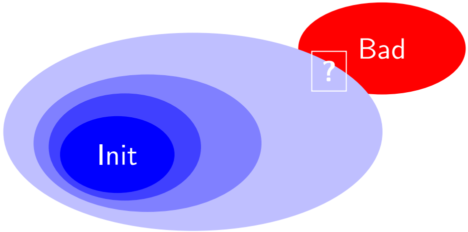


We note:

- $\mathrm{Post}(X) = \{x \mid \exists x' \in X \text{ s.t. } x \rightarrow x'\}$.
- $\mathrm{Post}^i(X) = \mathrm{Post}(\mathrm{Post}(\cdots \mathrm{Post}(X)))$.

Reachability Problem

We consider the following problem:

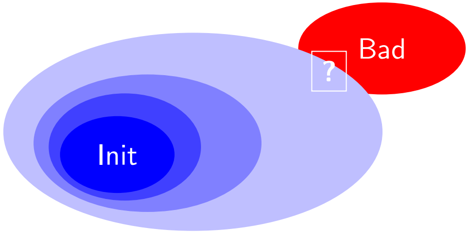


We note:

- $\mathrm{Post}(X) = \{x \mid \exists x' \in X \text{ s.t. } x \to x'\}$.
- $\mathrm{Post}^i(X) = \mathrm{Post}(\mathrm{Post}(\cdots \mathrm{Post}(X)))$.
- $\mathrm{Post}^*(X) = \bigcup_{i \geq 0} \mathrm{Post}^i(X)$.

Reachability Problem

We consider the following problem:

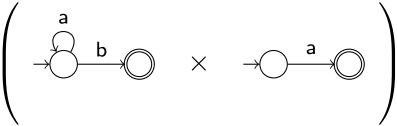


We note:

- $\mathrm{Post}(X) = \{x \mid \exists x' \in X \text{ s.t. } x \rightarrow x'\}$.
- $\mathrm{Post}^i(X) = \mathrm{Post}(\mathrm{Post}(\cdots \mathrm{Post}(X)))$.
- $\mathrm{Post}^*(X) = \bigcup_{i \geq 0} \mathrm{Post}^i(X)$. UNDECIDABLE (semi-algorithm)

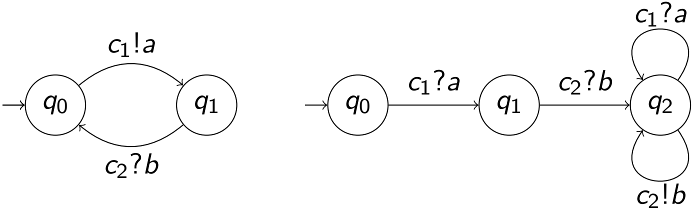
Representing Sets of Configurations

- We need to represent possibly infinite sets of configurations.
- We associate to each tuple of states of the CFSM a set of finite state automata (FUDFA) over Σ.
- The set of configurations corresponds to the (regular) language associated to each state.
- Ex: $\langle q_1, q_2 \rangle +$



represents the set of configurations $\langle q_1, q_2, a^* b, a \rangle$.

Complete example



$\langle q_0, q_0\rangle$		$\langle q_0, q_1\rangle$	
$\langle q_0, q_2\rangle$		$\langle q_1, q_0\rangle$	
$\langle q_1, q_1\rangle$		$\langle q_1, q_2\rangle$	

Basic Algorithm

- Input: CFSMs $\mathcal{M}_i = (Q_i, q_0, C_i, \Sigma_i, A_i, \delta_i)$ for $i \in \{1, \ldots, n\}$.
- Suppose that
  - $S \subseteq Q_1 \times \cdots \times Q_n$ is a set of states to explore (ex: $S = \{\langle q_0, \cdots, q_0 \rangle\}$),
  - F associates to each $s \in Q_1 \times \cdots \times Q_n$ a FUDFA.

Naive semi-algorithm

While $S \neq \emptyset$ **do**

Choose and remove some $s \in S$

For all possible transition $s \xrightarrow{\mathrm{op}} s'$

Compute $\mathrm{op}(F[s])$ as the effect the transition on $F[s]$

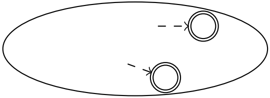
If $\mathrm{op}(F[s]) \not\subseteq F[s']$ **then**

$S := S \cup \{s'\}$

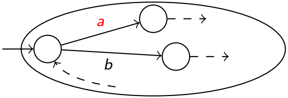
$F[s'] := F[s'] \cup \mathrm{op}(F[s])$.

Transformations needed

- Add a letter ($!a$):



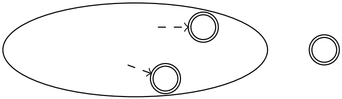
- Remove a letter ($?a$):



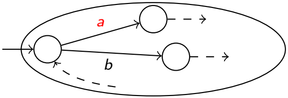
- Nothing to do with internal actions.
- Generalisation to sequences: just iterate!

Transformations needed

- Add a letter (!a):



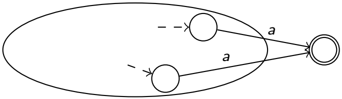
- Remove a letter (?a):



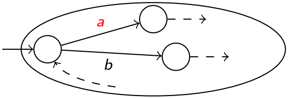
- Nothing to do with internal actions.
- Generalisation to sequences: just iterate!

Transformations needed

- Add a letter (!a):



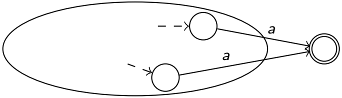
- Remove a letter (?a):



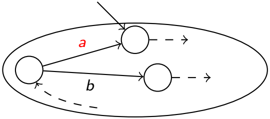
- Nothing to do with internal actions.
- Generalisation to sequences: just iterate!

Transformations needed

- Add a letter (!a):



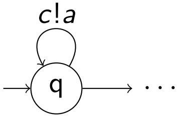
- Remove a letter (?a):



- Nothing to do with internal actions.
- Generalisation to sequences: just iterate!

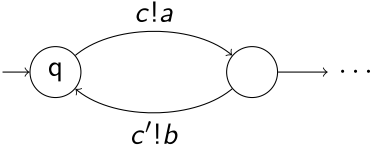
How to improve convergence?

- FUDFA allows to compute directly the result of infinitely iterating some cycles:



$$\rightsquigarrow \quad \langle q, a^* \rangle$$

- Pb: Cycles can induce non-regular sets of queue contents:



$$\rightsquigarrow \quad \langle q, a^n, b^n \rangle$$

- Need for characterization of accelerable loops.

Algorithm with accelerations

Improved semi-algorithm

While $S \neq \emptyset$ **do**

Choose and remove some $s \in S$

For all cycle θ from s

If $\mathrm{Adm}(\theta)$ then

Compute $\theta(F[s])$ as the effect of θ^* on $F[s]$

If $\theta(F[s]) \not\subseteq F[s']$ then $S := S \cup \{s'\}$.

For all possible transition $s \xrightarrow{\mathrm{op}} s'$

...

- Additional functions needed:
  - Research and selection of cycles,
  - Computation of acceleration.

Cycle selection and acceleration

- All the material needed can be adapted from Boigelot's thesis.
  - exact characterisation of accelerable cycles,
  - computation of the acceleration.
- For every sequence of operations σ,
  - $\sharp_!(\sigma)$ is the number of send operations,
  - $\sharp_?(\sigma)$ is the number of receive operations.
- A sequence involving only one queue is counting iff
  - $|\Sigma| = 1$ and $\sharp_!(\theta) > \sharp_?(\theta)$,
  - $|\Sigma| > 1$ and $\sharp_!(\theta) > 0$.
- Given a system with queues $\{c_1, \ldots, c_n\}$ and a cycle θ, $\theta_{|i}$ is the sub-sequence of transitions manipulating c_i.

Fundamental Results [Boigelot 98]

- For systems with only one queue, the result is the following.

Theorem (Single-queue systems)

For every set of configurations X and cycle θ, the set $\mathrm{Post}_\theta^*(X)$ is FUDFA representable.

- The result for systems with several queues is more restrictive.

Theorem (Multi-queue systems)

For every set of configurations X and cycle θ, the set $\mathrm{Post}_\theta^*(X)$ is FUDFA representable iff there do not exist i and j s.t $\theta_{|i}$ and $\theta_{|j}$ are counting.

Implementation

- Algorithm implemented in JAVA.
- Input: A set of CFSMs sharing a set of channels: text format or graphical editor (eclipse plugin).
- Computes successively the set of reachable states step by step + acceleration (at each iteration).
- Halting condition: Violated safety condition or predefined parameter (number of iterations).
- Few expriments on large scale examples for the moment.

More details about the implementation

The algorithm follows strictly the method described:

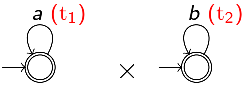
- We store the whole system in a transition table.
- Cycles:
    - we reseach elementary cycles only (research could be parametrerized),
    - non-counting cycles are added to the transition tables (meta transitions).
- A FUDFA is associated to each global state and the main loop of the algorithm can be executed.
- We use our own methods to handle the FUDFA.

Summary

- Modeling of unbounded communication queues (FIFO).
- Reachability algorithm based on:
  - Automata representation of queues,
  - Acceleration operations for selected cycles.
- Implementation of this algorithm into a prototype.

Future Work - Queue Manipulation

- In the current prototype:
  - Computing the set of states from which one can infinitely iterate a cycle.
  - Extend the tool to check linear temporal properties.
  - Improve data structure and algorithm.
- Adding counter in the queue representation [Bouajjani & Habermehl]



$\& \; t_1 = t_2 \quad \rightsquigarrow \quad \langle a^n, b^n \rangle$

$+$ New definition of acceleration.

- Considering more service policies.

Future Work - Verification of pNets

- Treating the other unbounded parameter.
  - Adding datas:
    - that can be finitely abstracted,
    - that can be represented by automata and combined with the current representation [Bardin et al].
  - Considering parameterized topologies.
- Defining a specification language for safety properties.