Formally Tracing Executions

From an Analysis Tool Back to a Domain Specific Modeling Language's Operational Semantics

Vlad Rusu and Laure Gonnord and Benoît Combemale

INRIA Lille/LIFL(Univ. Lille)

http://laure.gonnord.org/pro/
Laure.Gonnord@lifl.fr

Context - 1

- Model-based development.
- Design of new specialised languages : **Domain-Specific**.
- The syntax : meta-models (graphical grammars) : "**Modeling**"
- The (operational) semantics : methods of these objects.

► **Domain-Specific Modeling Languages** (DSML).

Context - 2

How do we deal with these DSMLs ?

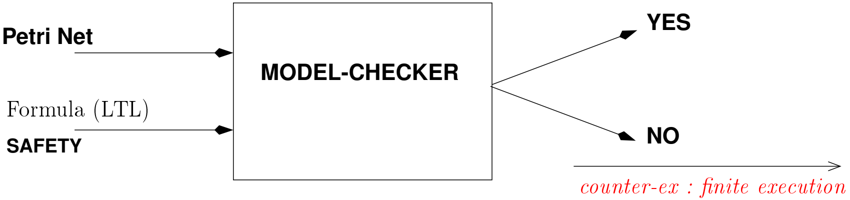
- Some of them are just modeling languages.
- Compilation through **model transformation** :
  - To other DSMLs.
  - To other “standard” languages : C, Lustre, . . .
- **Verification of** execution properties :
  - compilation into “classical” formal objects : Petri Nets, automata.
  - use an associated decision tool.

► The compilation produces a model which **semantics is well-defined**.

At the beginning - an example

[Combemale et al, Journal of Software, 2009]

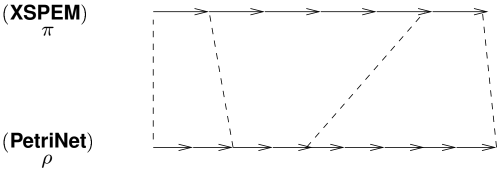
- A model transformation from **xSPEM** to **Petri Net** ;
- Both languages are designed in terms of metamodels ;
- After transformation, the resulted Petri Net is analysed.
  - ▶ **Model Checking**



ajouter animation avec une fleche qui remonte.

General Result

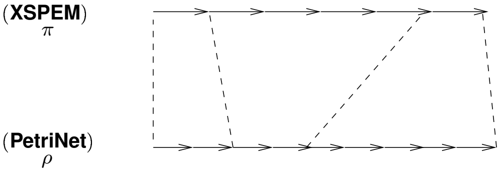
When Φ a **bisimulation** :



the result of the model checker **holds**.

General Result

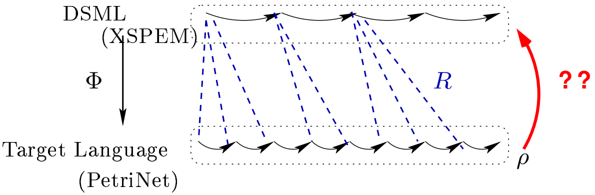
When Φ a **bisimulation** :



the result of the model checker **holds**.

- **But** how to deal with counter examples ?

The Problem



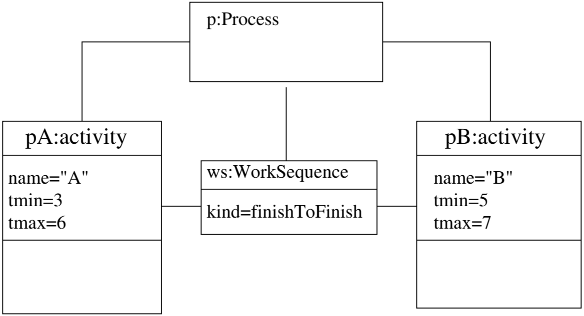
Given :

- A **relation** R between states ($|R| \leqslant |\Phi|$).
- An execution ρ of **the target language**,

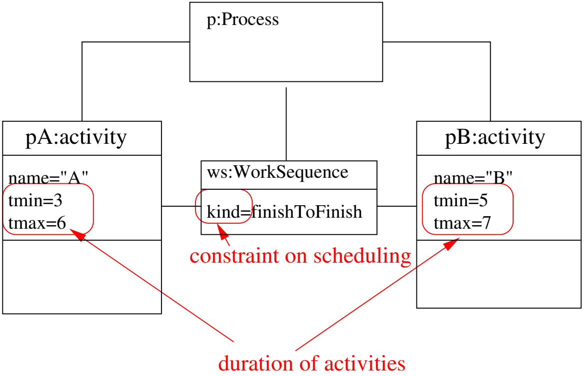
how expressing an execution in terms of **the input language** ?

▶ The backward tracing problem

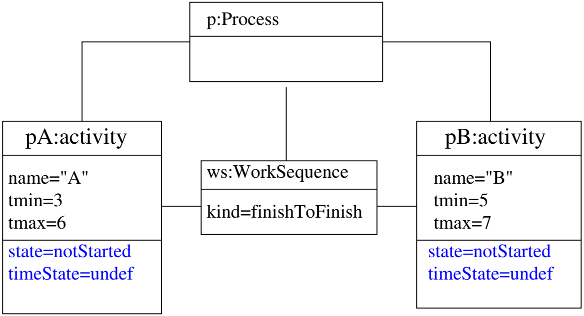
An example for Φ - Input Model (1/2)



An example for Φ - Input Model (1/2)

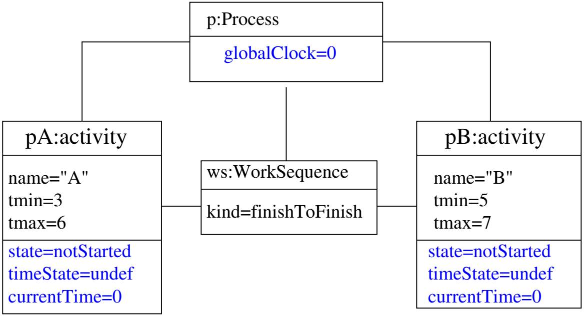


An example for Φ - Input Model (1/2)



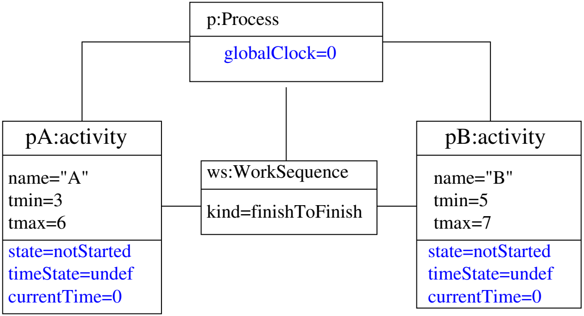
$state \in \{notStarted, started, finished\}$
$timeState \in \{undef, ok, tooLate, tooEarly\}$

An example for Φ - Input Model (1/2)



$state \in \{notStarted, started, finished\}$
$timeState \in \{undef, ok, tooLate, tooEarly\}$

An example for Φ - Input Model (1/2)



$state \in \{notStarted, started, finished\}$
$timeState \in \{undef, ok, tooLate, tooEarly\}$

evolving vars ▶ **operational semantics**

An example for Φ - Input Model (2/2)

Its operational semantics is a set of rules operating on *states* :

- The **global state** is $\{globalTime\} \times \Pi_{a \in A}(state_a, timeState_a, currentTime_a)$.

An example for Φ - Input Model (2/2)

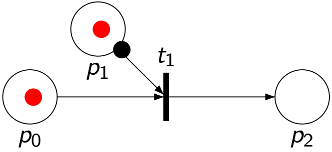
Its operational semantics is a set of rules operating on *states* :

- The **global state** is $\{globalTime\} \times \Pi_{a \in A}(state_a, timeState_a, currentTime_a)$.
- Given an activity a, the (non deterministic) evolution of its state is defined by **a set of rules** :
  - Evolution of $GlobalClock$ and all $currentTime$.
  - Evolution of $state_a$ (notstarted,started,finish) : must respect the static constraints $tmin, tmax$.
  - Evolution of $timeState_a$ (ok,tooLate,tooEarly), according to the real execution date of a).

► Implemented in Process.run() and Process.incTime() (they appear in the metamodel).

An example for Φ - Output Model

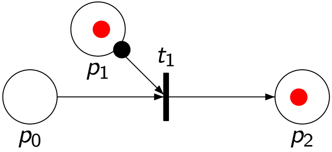
Classic PN



- A PrTPN **state** is $(marking, timestamp)$

An example for Φ - Output Model

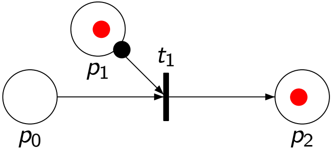
Classic PN



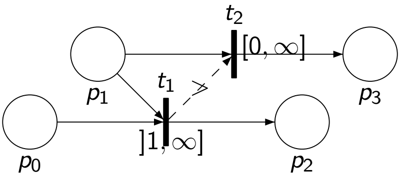
- A PrTPN **state** is $(marking, timestamp)$

An example for Φ - Output Model

Classic PN



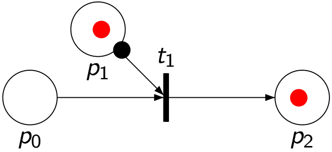
Prioritized TPN



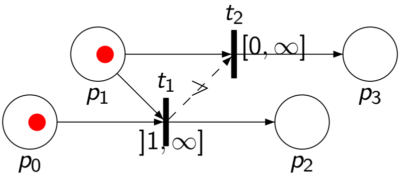
- A PrTPN **state** is $(marking, timestamp)$

An example for Φ - Output Model

Classic PN



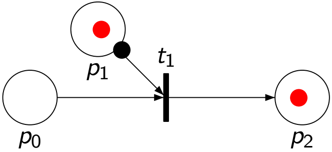
Prioritized TPN



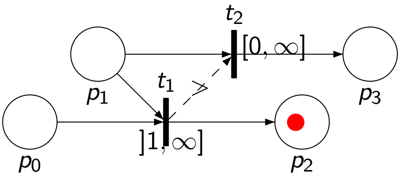
- A PrTPN **state** is $(marking, timestamp)$

An example for Φ - Output Model

Classic PN



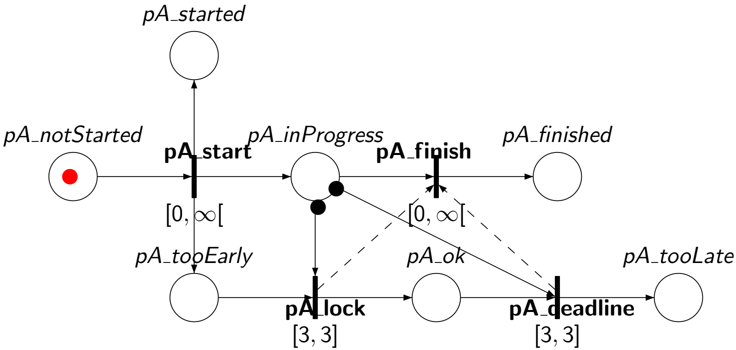
Prioritized TPN



- A PrTPN **state** is $(marking, timestamp)$

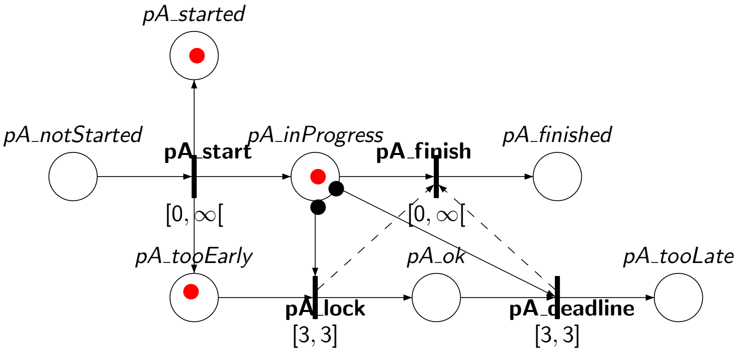
An example for Φ - Result

A Petri Net for each activity :



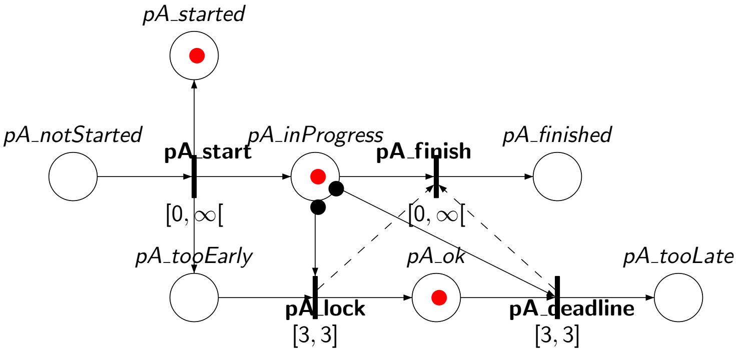
An example for Φ - Result

A Petri Net for each activity :



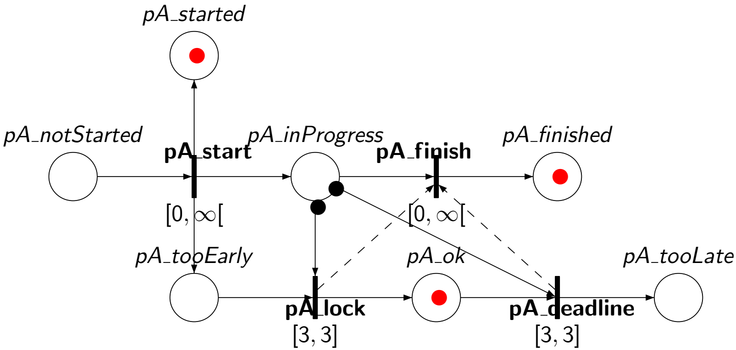
An example for Φ - Result

A Petri Net for each activity :



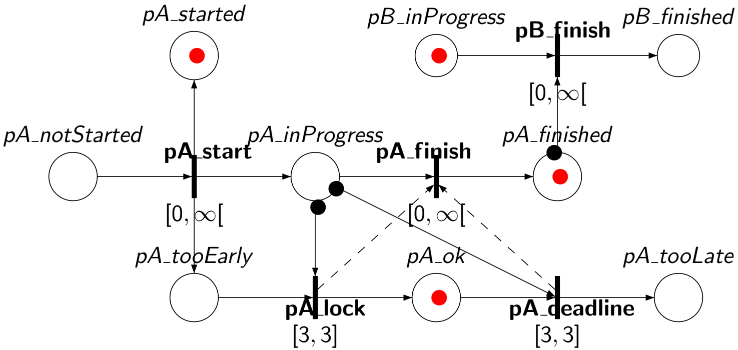
An example for Φ - Result

A Petri Net for each activity :



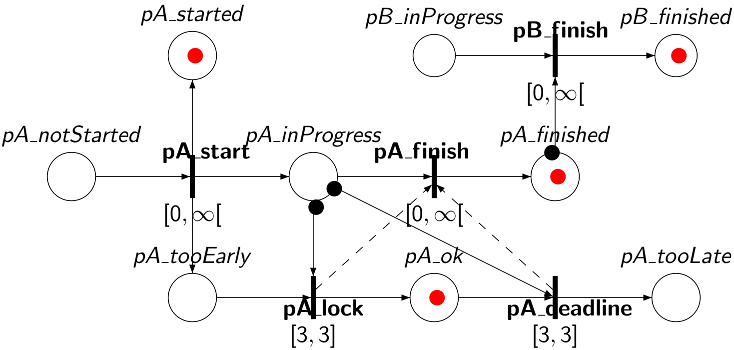
An example for Φ - Result

A Petri Net for each activity :



An example for Φ - Result

A Petri Net for each activity :



Relationship between states / executions

- A PN execution is a sequence of states : $(marking, timestamp)$.
- A XSPEM execution is a sequence of states $\{globalTime\} \times \Pi_{a \in A}(state_a, timeState_a, currentTime_a)$.

Relationship between states / executions

- A PN execution is a sequence of states : $(marking, timestamp)$.
- A XSPEM execution is a sequence of states $\{globalTime\} \times \Pi_{a \in A}(state_a, timeState_a, currentTime_a)$.

▶ Φ induces a **relation R** between states :

	XSPEM	**Petri Net**
	globalTime	time stamp
for all activity a example $a = pA$	$state_a = i$ $state_{pA} = started$	token in place a_i a token in place $pAStarted$
if a is finished example	$timeState_a = j$ $timeState_{pA} = ok$	token in place a_j (status) token in place $pAok$

An example for Φ - Analysis

Analysis of the output PN, with a LTL formula :

$$\Box \neg \left(pA\_finished \wedge pA\_ok \wedge pB\_finished \wedge pB\_ok \right)$$

An example for Φ - Analysis

Analysis of the output PN, with a LTL formula :

$$\Box\neg\,(pA\_finished \wedge pA\_ok \wedge pB\_finished \wedge pB\_ok)$$

TINA gives an execution ρ where both activities end in due time :

```
(m_0,0),pA_start,(m_1,0),pA_lock,(m_2,3),pA_finish,
(m_3,3),pB_start,(m_4,3),pB_lock,(m_5,8),pB_finish,
(m_6,8).
```

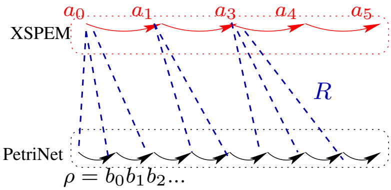
- A begins at $t = 0$ and finishes at $t = 3$.
- A begins at $t = 3$ and finishes at $t = 8$.

Our algorithm in summary

Given :

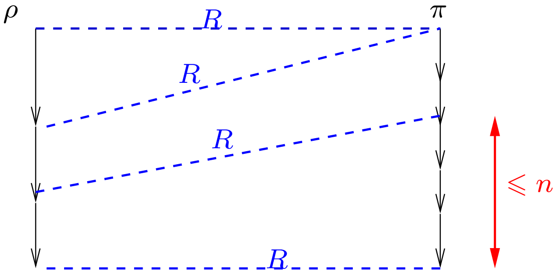
- **Input model :** syntax + implementation of the semantics.
- **Output model :** only the syntax.
- 3 other parameters : R (not necessarily a **simulation**), and n, and a ρ (execution of PN).

▶ our algorithm produces :



R-matching

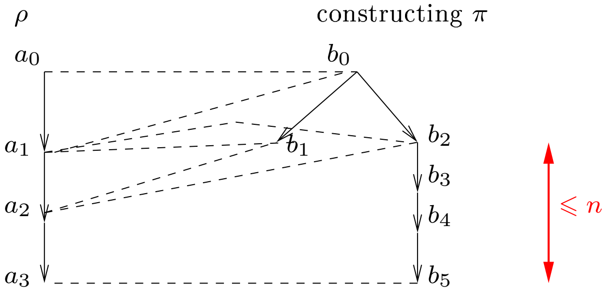
We consider **transition systems** with finite branching. Let R be a relation between states.



- The execution π (n,R) matches the execution ρ.

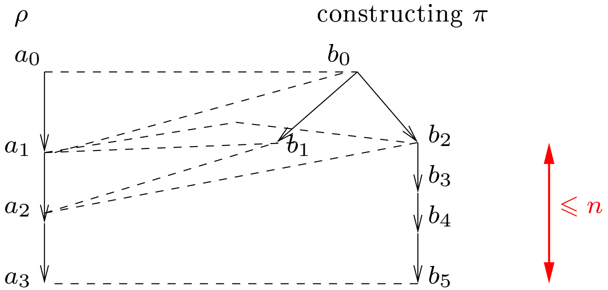
Algorithm

Trying to match ρ :



Algorithm

Trying to match ρ :



If the algorithms fails, R is **not** a simulation.

Algorithm - Result

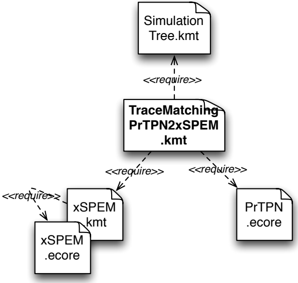
Theorem

Given ρ, the algorithm produces an execution that matches the longuest prefix that be (n, R) matched.

- Proof in the paper.

Implementation

- A generic implementation in Kermeta (Triskell).
- Instantiation on XSPEM to PetriNet.

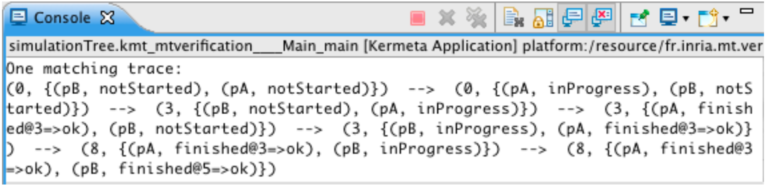


Implementation - usage

The user provides :

- the input model + semantics (+metamodel)
- an execution of the output model (here, given by TINA)
- a relation R (method), a bound n (here, 3).

and then :



Perpectives

- Implementation : genericity
- Algorithmic : **sharing** common parts in a model execution
- Theory : Relationship between Φ and R

Questions ?