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).
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

Petri Net

 Formula (LTL)

SAFETY

MODEL-CHECKER

counter-ex : finite execution

ajouter animation avec une fleche qui remonte.
General Result

When $\Phi$ a **bisimulation**:

\[(XSP\text{EM})_\pi\]
\[(\text{PetriNet})_\rho\]

the result of the model checker **holds**.
When $\Phi$ a \textbf{bisimulation}:

\[
\begin{align*}
(X_{\text{SEM}}) & \quad \pi \\
(PetriNet) & \quad \rho
\end{align*}
\]

the result of the model checker \textbf{holds}.

\begin{itemize}
    \item \textbf{But} how to deal with counter examples?
\end{itemize}
Given:

- A relation $R$ between states ($|R| \leq |\Phi|$).
- An execution $\rho$ of the target language.

How expressing an execution in terms of the input language?

- The backward tracing problem
1 Running Example
2 Formalisation and algorithm
3 Implementation and example
An example for $\Phi$ - Input Model (1/2)

```
tmax=6
tmin=3 kind=finishToFinish tmax=7
tmin=5
name="A" name="B"
pA:activity pB:activity
ws:WorkSequence
kind=finishToFinish
```

Rusu/Gonnord/Combemale (INRIA/LIFL) 2010, Oct
An example for $\Phi$ - Input Model (1/2)

- **duration of activities**: $t_{\text{max}}=6$, $t_{\text{min}}=3$, kind=finishToFinish
- **constraint on scheduling**: $t_{\text{max}}=7$, $t_{\text{min}}=5$
- **name**:
  - pA: activity
  - pB: activity
- **ws: WorkSequence**
  - kind=finishToFinish
  - $t_{\text{min}}=3$, $t_{\text{max}}=6$
  - $t_{\text{min}}=5$, $t_{\text{max}}=7$

**Notations and Concepts**
- **evolving vars**
- **operational semantics**
An example for $\Phi$ - Input Model (1/2)

- $p$: Process
  - $pA$: activity
    - name = "A"
    - $t_{min} = 3$
    - $t_{max} = 6$
    - state = notStarted
    - timeState = undef
  - $ws$: WorkSequence
    - kind = finishToFinish
  - $pB$: activity
    - name = "B"
    - $t_{min} = 5$
    - $t_{max} = 7$
    - state = notStarted
    - timeState = undef

- $state \in \{\text{notStarted, started, finished}\}$
- $timeState \in \{\text{undef, ok, tooLate, tooEarly}\}$
An example for \( \Phi \) - Input Model (1/2)

\[
\begin{align*}
\text{pA:activity} & \quad \text{ws:WorkSequence} & \quad \text{pB:activity} \\
\text{name} = "A" & \quad \text{kind} = \text{finishToFinish} & \quad \text{name} = "B" \\
\text{tmin} = 3 & \quad & \text{tmin} = 5 \\
\text{tmax} = 6 & \quad & \text{tmax} = 7 \\
\text{state} = \text{notStarted} & \quad & \text{state} = \text{notStarted} \\
\text{timeState} = \text{undef} & \quad & \text{timeState} = \text{undef} \\
\text{currentTime} = 0 & \quad & \text{currentTime} = 0
\end{align*}
\]

\( \text{state} \in \{\text{notStarted, started, finished}\} \)
\( \text{timeState} \in \{\text{undef, ok, tooLate, tooEarly}\} \)
An example for $\Phi$ - Input Model (1/2)

$state \in \{\text{notStarted, started, finished}\}$

$\text{timeState} \in \{\text{undef, ok, tooLate, tooEarly}\}$

 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 $\Phi$ - Input Model (2/2)

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

- The **global state** is
  \[
  \{\text{globalTime}\} \times \Pi_{a \in A}(\text{state}_a, \text{timeState}_a, \text{currentTime}_a).
  \]
- Given an activity $a$, the (non deterministic) evolution of its state is defined by a set of rules:
  - Evolution of $\text{GlobalClock}$ and all $\text{currentTime}$.
  - Evolution of $\text{state}_a$ (notstarted,started,finish) : must respect the static constraints $t_{min}, t_{max}$.
  - Evolution of $\text{timeState}_a$ (ok,tooLate,tooEarly), according to the real execution date of $a$.
An example for $\Phi$ - Output Model

A PrTPN state is $(marking, timestamp)$
An example for $\Phi$ - Output Model

A PrTPN state is $(marking, timestamp)$
An example for $\Phi$ - Output Model

**Classic PN**

- $p_0$ to $p_1$ to $p_2$
- $p_1$ is the only place with a token.

**Prioritized TPN**

- $p_0$ to $p_1$ with $t_1$ in $[1, \infty)$
- $p_1$ to $p_2$ with $t_2$ in $[0, \infty)$

- $p_0$ is the only place with a token.

▶ A PrTPN state is $(marking, timestamp)$
An example for $\Phi$ - Output Model

Classic PN

Prioritized TPN

▶ A PrTPN state is $(\text{marking}, \text{timestamp})$
An example for $\Phi$ - Output Model

**Classic PN**

![Classic PN Diagram]

**Prioritized TPN**

![Prioritized TPN Diagram]

▶ A PrTPN state is \((\text{marking}, \text{timestamp})\)
An example for $\Phi$ - Result

A Petri Net for each activity:

- $p_{A\_notStarted}$
- $p_{A\_started}$
- $p_{A\_inProgress}$
- $p_{A\_finish}$
- $p_{A\_finished}$
- $p_{A\_tooEarly}$
- $p_{A\_ok}$
- $p_{A\_tooLate}$
- $p_{A\_lock}$
- $p_{A\_deadline}$
An example for $\Phi$ - Result

A Petri Net for each activity:

- $pA_{start}$
- $pA_{notStarted}$
- $pA_{inProgress}$
- $pA_{finish}$
- $pA_{finished}$
- $pA_{tooEarly}$
- $pA_{deadline}$
- $pA_{lock}$
- $pA_{tooLate}$

Transition labels:
- $[0, \infty]$ for $pA_{start}$, $pA_{finish}$, and $pA_{deadline}$
- $[0, \infty]$ for $pA_{lock}$
- $[3, 3]$ for $pA_{start}$, $pA_{inProgress}$, $pA_{lock}$, and $pA_{deadline}$

Rusu/Gonnord/Combemale (INRIA/LIFL) 2010, Oct
An example for $\Phi$ - Result

A Petri Net for each activity:

- $pA_{notStarted}$
- $pA_{started}$
- $pA_{tooEarly}$
- $pA_{inProgress}$
- $pA_{lock}$
- $pA_{ok}$
- $pA_{deadline}$
- $pA_{finish}$
- $pA_{tooLate}$
- $pA_{finished}$

Transitions:

- $pA_{start}$ to $pA_{inProgress}$ with $\mathbb{Z}^+$
- $pA_{finish}$ to $pA_{tooEarly}$ with $\mathbb{Z}^+
- pA_{lock}$ to $pA_{ok}$ with $[3, 3]\times[3, 3]
- pA_{ok}$ to $pA_{deadline}$ with $[3, 3]\times[3, 3]
- pA_{tooEarly}$ to $pA_{ok}$ with $[3, 3]\times[3, 3]

Places:

- $pA_{notStarted}$
- $pA_{inProgress}$
- $pA_{lock}$
- $pA_{ok}$
- $pA_{deadline}$
- $pA_{finish}$
- $pA_{tooEarly}$
- $pA_{tooLate}$
- $pA_{finished}$
An example for $\Phi$ - Result

A Petri Net for each activity:

\begin{itemize}
    \item $p_{A\_started}$
    \item $p_{A\_notStarted}$
    \item $p_{A\_start}$
    \item $p_{A\_inProgress}$
    \item $p_{A\_finish}$
    \item $p_{A\_finished}$
    \item $p_{A\_tooEarly}$
    \item $p_{A\_ok}$
    \item $p_{A\_tooLate}$
    \item $p_{A\_lock}$
    \item $p_{A\_deadline}$
\end{itemize}
An example for $\Phi$ - Result

A Petri Net for each activity:

- For activity A:
  - $pA_{start}$
  - $pA_{notStarted}$
  - $pA_{notStarted}$
  - $pA_{finish}$
  - $pA_{lock}$
  - $pA_{tooEarly}$
  - $pA_{ok}$
  - $pA_{tooLate}$

- For activity B:
  - $pB_{inProgress}$
  - $pB_{finish}$
  - $pB_{finished}$

Transitions:
- $\text{start} \rightarrow pA_{start}$
- $pA_{notStarted} \rightarrow pA_{start}$
- $pA_{start} \rightarrow pA_{inProgress}$
- $pA_{inProgress} \rightarrow pA_{finish}$
- $pA_{finish} \rightarrow pA_{lock}$
- $pA_{lock} \rightarrow pA_{start}$
- $pA_{tooEarly} \rightarrow pA_{finish}$
- $pA_{ok} \rightarrow pA_{finish}$
- $pA_{tooLate} \rightarrow pA_{finished}$
An example for $\Phi$ - Result

A Petri Net for each activity:

- $p_{A\_notStarted}$
- $p_{A\_start}$
- $p_{A\_inProgress}$
- $p_{A\_finish}$
- $p_{A\_tooEarly}$
- $p_{A\_lock}$
- $p_{A\_deadline}$
- $p_{A\_ok}$
- $p_{A\_tooLate}$
- $p_{B\_inProgress}$
- $p_{B\_finish}$
- $p_{B\_finished}$

Transition rates:
- $[0, \infty]$ for $p_{A\_start}$ to $p_{A\_inProgress}$
- $[0, \infty]$ for $p_{B\_finish}$ to $p_{B\_finished}$
- $[3, 3]$ for $p_{A\_lock}$ to $p_{A\_deadline}$

Rusu/Gonnord/Combemale (INRIA/LIFL) 2010, Oct
A PN execution is a sequence of states:

\((\text{marking, timestamp})\).

A XSPEM execution is a sequence of states

\(\{\text{globalTime}\} \times \Pi_{a \in A}(\text{state}_a, \text{timeState}_a, \text{currentTime}_a)\).
Running Example

# Relationship between states / executions

- A PN execution is a sequence of states: \((\text{marking, timestamp})\).
- A XSPEM execution is a sequence of states: 
  \(\{\text{globalTime}\} \times \Pi_{a \in A}(\text{state}_a, \text{timeState}_a, \text{currentTime}_a)\).

Φ induces a relation \(R\) between states:

<table>
<thead>
<tr>
<th>XSPEM</th>
<th>Petri Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{globalTime}</td>
<td>\text{time stamp}</td>
</tr>
<tr>
<td>\text{state}_a = i</td>
<td>\text{token in place } a_i</td>
</tr>
<tr>
<td>\text{state}_{pA} = \text{started}</td>
<td>\text{a token in place } pA_{Started}</td>
</tr>
<tr>
<td>\text{timeState}_a = j</td>
<td>\text{token in place } a_j (\text{status})</td>
</tr>
<tr>
<td>\text{timeState}_{pA} = \text{ok}</td>
<td>\text{token in place } pA_{ok}</td>
</tr>
</tbody>
</table>
An example for $\Phi$ - Analysis

Analysis of the output PN, with a LTL formula:

$$\square \neg(pA_{\text{finished}} \land pA_{\text{ok}} \land pB_{\text{finished}} \land pB_{\text{ok}})$$
An example for $\Phi$ - Analysis

Analysis of the output PN, with a LTL formula:

$$\Box \neg (pA_{\text{finished}} \land pA_{\text{ok}} \land pB_{\text{finished}} \land pB_{\text{ok}})$$

TINA gives an execution $\rho$ where both activities end in due time:

$$(m_0,0), pA_{\text{start}}, (m_1,0), pA_{\text{lock}}, (m_2,3), pA_{\text{finish}},$$

$$(m_3,3), pB_{\text{start}}, (m_4,3), pB_{\text{lock}}, (m_5,8), pB_{\text{finish}},$$

$$(m_6,8).$$

- 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 $\rho$ (execution of PN).

▶ our algorithm produces:

\[
\begin{align*}
R & = b_0 b_1 b_2 \ldots \\
\rho & = a_0 a_1 a_2 \ldots
\end{align*}
\]
1. Running Example

2. Formalisation and algorithm

3. Implementation and example
R-matching

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

The execution $\pi \ (n,R)$ matches the execution $\rho$. 
Algorithm

Trying to match $\rho$:

$\rho$

constructing $\pi$

$a_0$

$b_0$

$a_1$

$b_1$

$a_2$

$b_2$

$a_3$

$b_3$

$b_4$

$b_5$

$\leq n$
Algorithm

Trying to match $\rho$:

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

Given $\rho$, the algorithm produces an execution that matches the longest prefix that be $(n, R)$ matched.

▶ Proof in the paper.
1 Running Example

2 Formalisation and algorithm

3 Implementation and example
Implementation

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

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:

```
simulationTree.kmt_mtverification__Main__main [Kermeta Application] platform:/resource/fr.inria.mt.ver
One matching trace:
(0, {(pB, notStarted), (pA, notStarted)}) -->(0, {(pA, inProgress), (pB, notStarted)}) -->(3, {(pB, notStarted), (pA, inProgress)}) -->(3, {(pA, finished@3=>ok), (pB, notStarted)}) -->(3, {(pB, inProgress), (pA, finished@3=>ok)}) -->(8, {(pA, finished@3=>ok), (pB, inProgress)}) -->(8, {(pA, finished@3=>ok), (pB, finished@5=>ok)})
```
Perpectives

- Implementation: genericity
- Algorithmic: sharing common parts in a model execution
- Theory: Relationship between $\Phi$ and $R$
Questions?