CosyVerif: An Open Source Extensible Verification Environment

Étienne André, Benoît Barbot, Clément Démoulins, Lom Hillah, Francis Hulin-Hubard, Fabrice Kordon, Y. Lembachar, A. Linard, Laure Petrucci

ENS Cachan, Université Paris 6, Université Paris 13, Epita (France)

11th December 2014
Motivation (1/2)

Many tools for distributed systems verification

- Relying on different formalisms
- Solving different problems
- Applying formal methods to dedicated cases studies
- Running on different OS
- Requiring some difficult installation procedures
- Technological transfer to industry uneasy
Motivation (2/2)

Needs for:

- A *unified representation*
- Coordinated effort to better handle a complex context of interrelated formal notations
  - Variants of Petri nets
  - Variants of automata
  - etc.
- **Interoperability** and tool integration

- For developers: Easy integration of tools
- For users: Easy installation and use
Outline

1. A Unified Representation of Formal Notations
2. The CosyVerif Environment
3. Tools Integrated in CosyVerif
4. Summary and Evolutions
A Unified Representation of Formal Notations

- FML: Formalism Markup Language
- GrML: Graph Markup Language
- Architecture of Formalisms in CosyVerif
- Automated Compliance Checking
- Good Practices to Create New Formalisms

The CosyVerif Environment

Tools Integrated in CosyVerif

Summary and Evolutions
A Unified Representation

Needs to define formalisms

- Easy definition and share of formalisms
- It is crucial to define formalisms in an easy and sustainable way
- It is crucial to share portions of formalisms (e.g. inheritance)
- It is crucial to combine portions of formalisms (e.g. modularity and hierarchy)
A two-layered XML-based modelling language

*Meta-Meta*  
FML

is an instance of

*Meta*  
User Formalism

specialises

GrML

*Instance*  
User Model

is an instance of

complies with

is structured by

Étienne André (Paris 13)  
The CosyVerif Platform  
11th December 2014
FML: Formalism Markup Language

- **Rule**
  - Formalism
    - name
    - abstract: [0..1] = false
  - LeafAttribute
    - name
    - defaultValue: [0..1]
    - refType: [0..1]
  - ComplexAttribute
    - name
    - refType: [0..1]
    - combineChild: interleave | choice
      - [0..1] = interleave
  - Child
    - refName
    - minOccurs: [0..1] = 1
    - maxOccurs: [0..1] = ∞
  - xi:Include
    - href
  - *
  - *

- **NodeType**
  - name

- **ArcType**
  - name

- **Ref**
  - href
  - minOccurs: [0..1] = 0
  - maxOccurs: [0..1] = ∞
Example: The Automata Formalism

```xml
<formalism name="Automaton" xmlns="http://cosyverif.org/ns/formalism">
  <leafAttribute name="initialState" />
  <leafAttribute name="finalState" />
  <complexType name="type" refType="state">
    <child refName="initialState" minOccurs="0" maxOccurs="1"/>
    <child refName="finalState" minOccurs="0" maxOccurs="1"/>
  </complexType>
  <leafAttribute name="name" refType="state"/>
  <leafAttribute name="label" refType="transition"/>
  <nodeType name="state"/>
  <arcType name="transition"/>
</formalism>
```
GrML: Graph Markup Language
Example: An Automaton

Example of a GrML code:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<model formalismUrl="http://formalisms.cosyverif.org/graph.fml" xmlns="http://cosyverif.org/ns/model">
  <node id="1" nodeType="vertex">
    <attribute name="name">u</attribute>
  </node>
  <node id="2" nodeType="vertex">
    <attribute name="name">v</attribute>
  </node>
  <arc id="101" arcType="transition" source="1" target="2"/>
  <arc id="102" arcType="transition" source="2" target="1"/>
</model>
```

Corresponding automaton:

```
```

u \rightarrow v
Architecture of Formalisms

Application to Automata and Petri Nets

- Automaton
- Expressions and Boolean expressions
- Abstract timed automaton
- Abstract PN-Core
- Symmetric-Net
- P/T Net
- abstract PN-Modules
- Hybrid Automaton
- Abstract parametric timed automaton
- Linear Hybrid Automaton
- Abstract parametric timed automaton
- Stopwatch Automaton
- Parametric Stopwatch Automaton
- Timed Automaton
- Stochastic-Net
- Hierarchical Place/Transition-Net
- Parametric Timed Automaton
- Symmetric-Net with-Bags

Etienne André (Paris 13)
Automated Compliance Checking

- GrML and FML syntaxes validated using RELAX-NG

- Compliance of a GrML model against its FML formalism
  - Schematron rules derived from the XML description of the formalism
  - Particular constraints such as “in a Petri net formalism, no arc should connect two nodes of the same type”
  - Implemented in GrML-Check
Good Practices for Defining Formalisms

- **Abstract vs. concrete formalisms**
  - Similarity with object-oriented paradigm
  - Definite concrete formalisms as late as possible in the hierarchy
  - Only concrete formalisms can be instantiated

- **Separate and reuse**
  - Separate data types (integers, colors, etc.)
  - Example in our hierarchy: Expressions and Boolean expression

- **Progress by small increments**
  - P/T nets extended to Stochastic nets
Outline

1. A Unified Representation of Formal Notations
2. The CosyVerif Environment
3. Tools Integrated in CosyVerif
4. Summary and Evolutions
**CosyVerif**, a new integration platform [AHHKLLPP13]

- Based on Web services
- Easy integration of tools (into services)

- Relies on **FML** and **GrML**

- **Advantages**
  - Unified model representation
  - Easy addition of new formalisms

- **Applications: set of formalisms**
  - Large family of (timed) automata and Petri nets
Easy to install: simple and light multi-platform client connecting to servers
- Tool invocation through Web services transparent to the end-user
- Cloud-based architecture
1. A Unified Representation of Formal Notations
2. The CosyVerif Environment
3. Tools Integrated in CosyVerif
4. Summary and Evolutions
Tools Integration in CosyVerif

- One official client: Coloane (platform-independent)
  - (Command-line client also available)

- Multiple graph-based formalisms supported
  - Petri nets and extensions
  - Hybrid automata
  - (Parametric) timed automata
  - ... and more!

- Numerous tools integrated to CosyVerif
  - More than 11 tools and even more integrated services
  - Integration via Web services: easy to use and compose
# Formalisms and Tools

<table>
<thead>
<tr>
<th>Formalisms</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Petri Nets</strong></td>
<td>PROD (Univ. Helsinki, Symmetric nets)</td>
</tr>
<tr>
<td></td>
<td>PNXDD (LIP6, Symmetric nets)</td>
</tr>
<tr>
<td></td>
<td>Crocodile (LIP6, Symmetric nets w. bags)</td>
</tr>
<tr>
<td></td>
<td>Cunf (LSV, P/T nets)</td>
</tr>
<tr>
<td></td>
<td>Cosmos (LSV, Stochastic Petri nets)</td>
</tr>
<tr>
<td></td>
<td>GreatSPN invariants (Univ. Torino, P/T nets)</td>
</tr>
<tr>
<td></td>
<td>Structural bounds (LIP6, P/T nets)</td>
</tr>
<tr>
<td></td>
<td>Unfold into P/T nets (LIP6, Symmetric nets)</td>
</tr>
<tr>
<td></td>
<td>Various exports (LIP6, P/T nets)</td>
</tr>
<tr>
<td><strong>Automata</strong></td>
<td>IMITATOR (LIPN, Timed automata)</td>
</tr>
<tr>
<td></td>
<td>Modgraph (LIPN, Synchronised automata)</td>
</tr>
</tbody>
</table>

- PROD (Univ. Helsinki, Symmetric nets) [Kordon et al., 2012]
- PNXDD (LIP6, Symmetric nets) [Colange et al., 2011]
- Crocodile (LIP6, Symmetric nets w. bags) [Baldan et al., 2012]
- Cunf (LSV, P/T nets) [Ballarini et al., 2011]
- Cosmos (LSV, Stochastic Petri nets) [Ballarini et al., 2011]
- GreatSPN invariants (Univ. Torino, P/T nets) [Ballarini et al., 2011]
- Structural bounds (LIP6, P/T nets) [Ballarini et al., 2011]
- Unfold into P/T nets (LIP6, Symmetric nets) [Ballarini et al., 2011]
- Various exports (LIP6, P/T nets) [Ballarini et al., 2011]
- IMITATOR (LIPN, Timed automata) [André et al., 2012]
- Modgraph (LIPN, Synchronised automata) [Lakos and Petrucci, 2004]
Two bundles available (http://download.cosyverif.org)

All services  Petri Nets
Statistical model checker [Ballerini et al., 2011]

- Input: Generalised Stochastic Petri Nets with general distribution (GSPN) and a Hybrid Automaton Stochastic Logic (HASL) formula
- Output: Statistical estimation of the formula with a confidence interval
State space generation & CTL verification [Colange et al., 2011]

- Symbolic/symbolic approach based on Symmetric Nets with Bags [Haddad et al., 2009]

- Two symbolic techniques to counter state space explosion
  1. symmetries to reduce the reachability graph [Chiola et al., 1991]
  2. hierarchical Set Decision Diagrams to store the reachability graph [Couvreur and Thierry-Mieg, 2005]
Unfolding-based verification of Petri nets with read arcs (contextual nets) [Baldan et al., 2012]

- **Features**
  - Unfolding construction tool [Rodríguez et al., 2011]
  - Reachability and deadlock checking tool [Rodríguez and Schwoon, 2012]

- **Characteristics**
  - Unfoldings fully represent the state space of a c-net by a partial order rather than by a set of interleavings
    - Often exponentially smaller than the state space, and never larger
  - c-net unfoldings can be exponentially more compact than those of corresponding Petri nets [Baldan et al., 2012]
Parameter synthesis for real-time systems [André et al., 2012]

- Quantitative robustness analysis
  - “Can we increase some of the timing delays such that the system still behaves well?”
- Schedulability analysis
- Hybrid system verification
Construction and analysis of **modular state spaces**

[Lakos and Petrucci, 2004]

- Modular State Spaces for Synchronised Automata
  - synchronisation structure
  - only reachable parts of the automata

- Analysis
  - forward and backward reachability
  - deadlock-checking
  - liveness
State space generation and CTL formulæ evaluation on P/T nets [Hong et al., 2012]

- Handles Symmetric Nets through their unfolding into an equivalent P/T net
- Exploits hierarchy: a state is seen as a tree, where the leaves correspond to place markings
- Relies on Set Decision Diagrams [Couvreur and Thierry-Mieg, 2005]
1 A Unified Representation of Formal Notations
2 The CosyVerif Environment
3 Tools Integrated in CosyVerif
4 Summary and Evolutions
An Open Environment

- Entirely open source (GPL/AGPL)
  - Server
  - Client(s)
  - Bundles
  - Tools

- Open to contributions
  - Tool integration
  - Alternative clients
  - New formalisms

- A repository of models using a common syntax
  - Coming from the integrated tools, and the model checking contests
    [Kordon et al., 2013]
Recent and Ongoing Evolutions

- **Asynchronous tool invocation**
  - Get the result later (e.g., by email)

- **Federation of servers and use of clusters**
  - Enable load balancing

- **Repository** of formalisms and models

- **Command-line** version of the underlying platform
Future Evolutions

- **Enhanced interaction** between tools
  - Output of a tool as input of another one

- Handling **semantics** (bridges between formalisms)
  - Also allows system simulation

- Handling **heterogeneous models** (mixing different formalisms)
Future Evolutions

- **Enhanced interaction** between tools
  - Output of a tool as input of another one
- Handling **semantics** (bridges between formalisms)
  - Also allows system simulation
- Handling **heterogeneous models** (mixing different formalisms)

Try it!

http://cosyverif.org/
Bibliography
References I


Licensing
Title: Scottish Kitten
Author: RN3DLL
Source: https://commons.wikimedia.org/wiki/File:Scottish_Kitten.png
License: CC BY-SA 3.0
This presentation can be published, reused and modified under the terms of the Creative Commons license Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0)

Author: The CosyVerif team

https://creativecommons.org/licenses/by-sa/3.0/