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

CosyVerif

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

Outline

1. 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
2. The CosyVerif Environment
3. Tools Integrated in CosyVerif
4. Summary and Evolutions

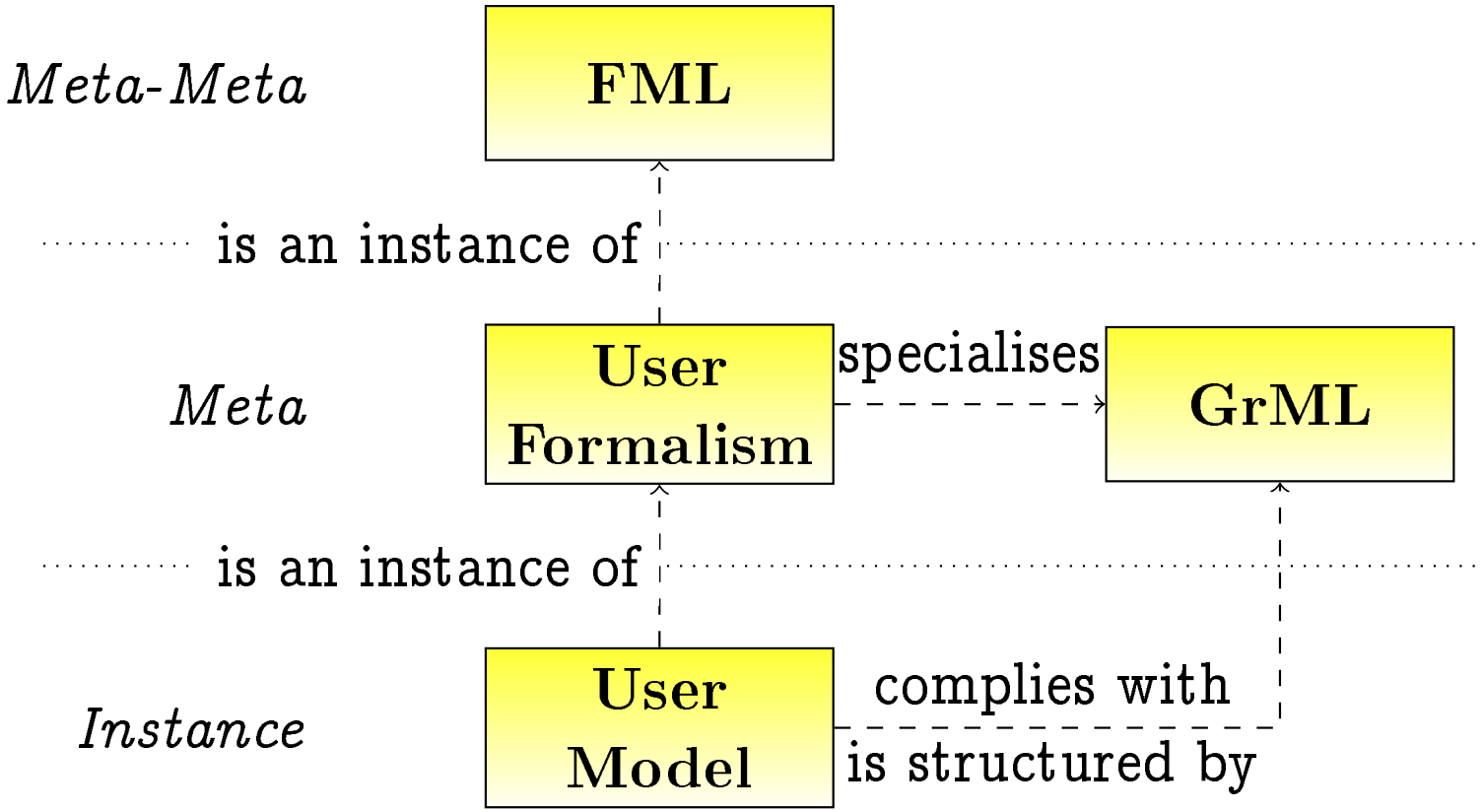
A Unified Representation

Needs to define formalisms

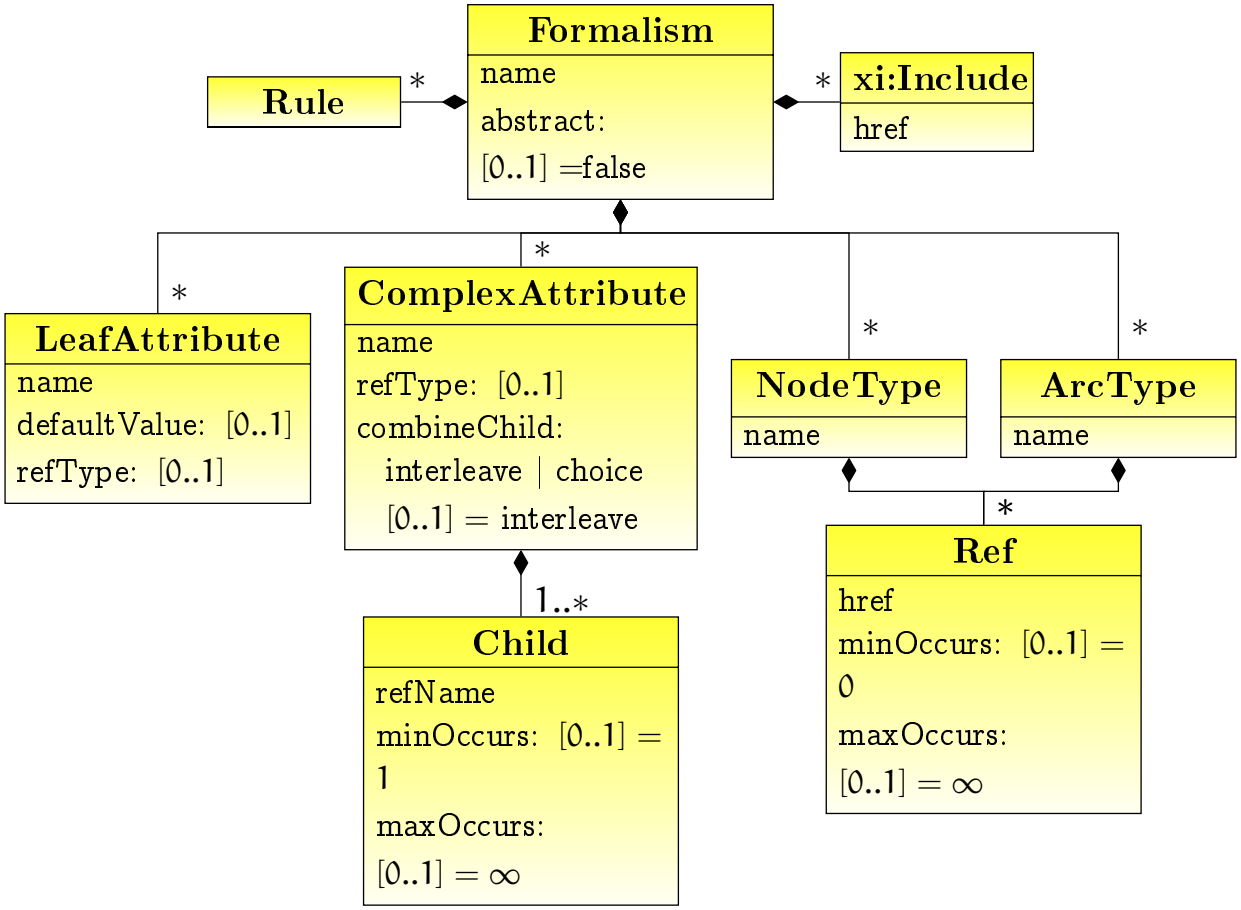
- Easy definition and share of formalisms

- 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



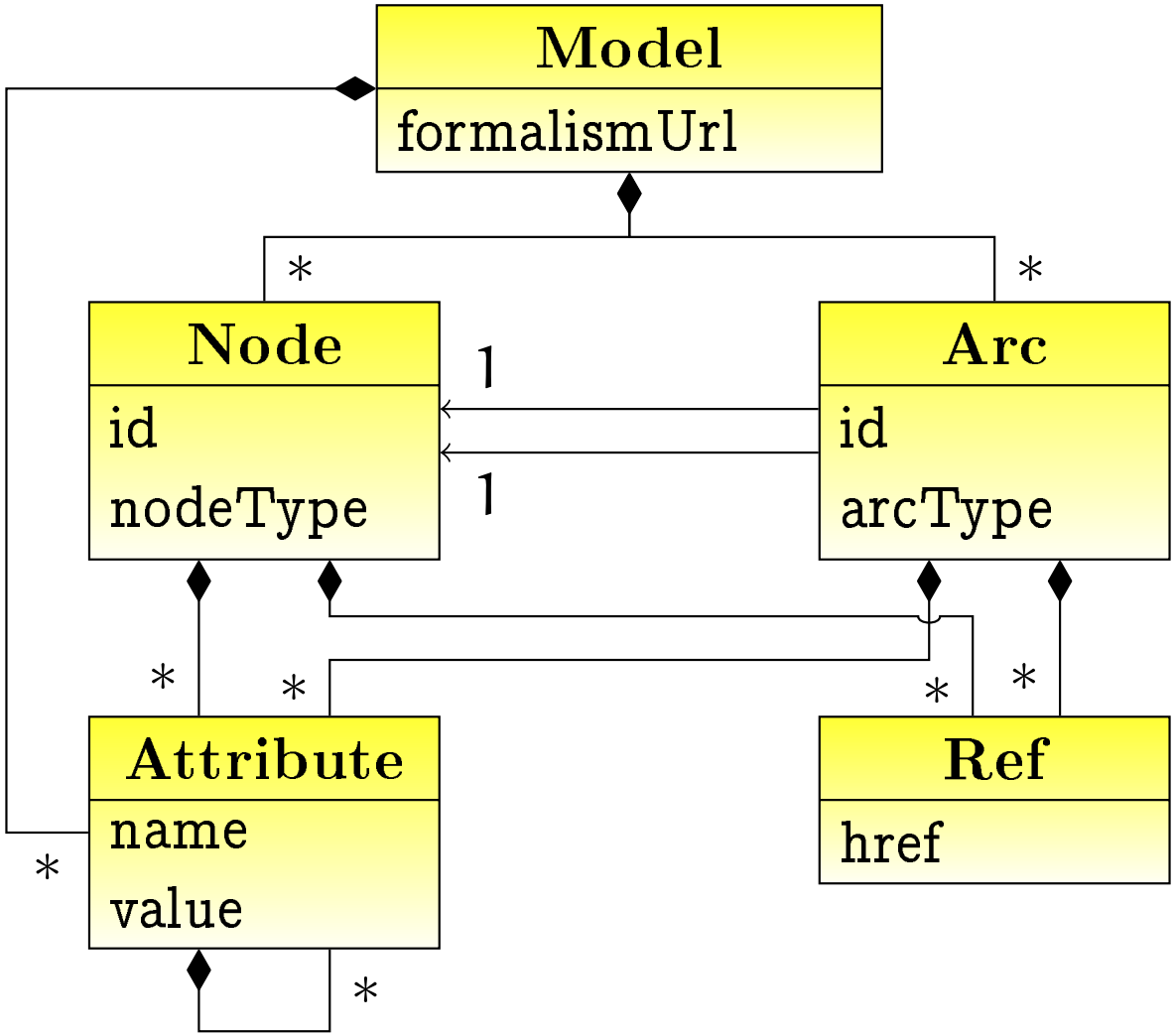
FML: Formalism Markup Language



Example: The Automata Formalism

```
<formalism name="Automaton" xmlns="http://cosyverif.org/ns/forma
    <leafAttribute name="initialState" />
    <leafAttribute name="finalState" />
    <complexAttribute name="type" refType="state">
        <child refName="initialState" minOccurs="0" maxOccurs="1
        <child refName="finalState" minOccurs="0" maxOccurs="1"/
    </complexAttribute>
    <leafAttribute name="name" refType="state"/>
    <leafAttribute name="label" refType="transition"/>
    <nodeType name="state"/>
    <arcType name="transition"/>
</formalism>
```

GrML: Graph Markup Language

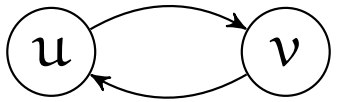


Example: An Automaton

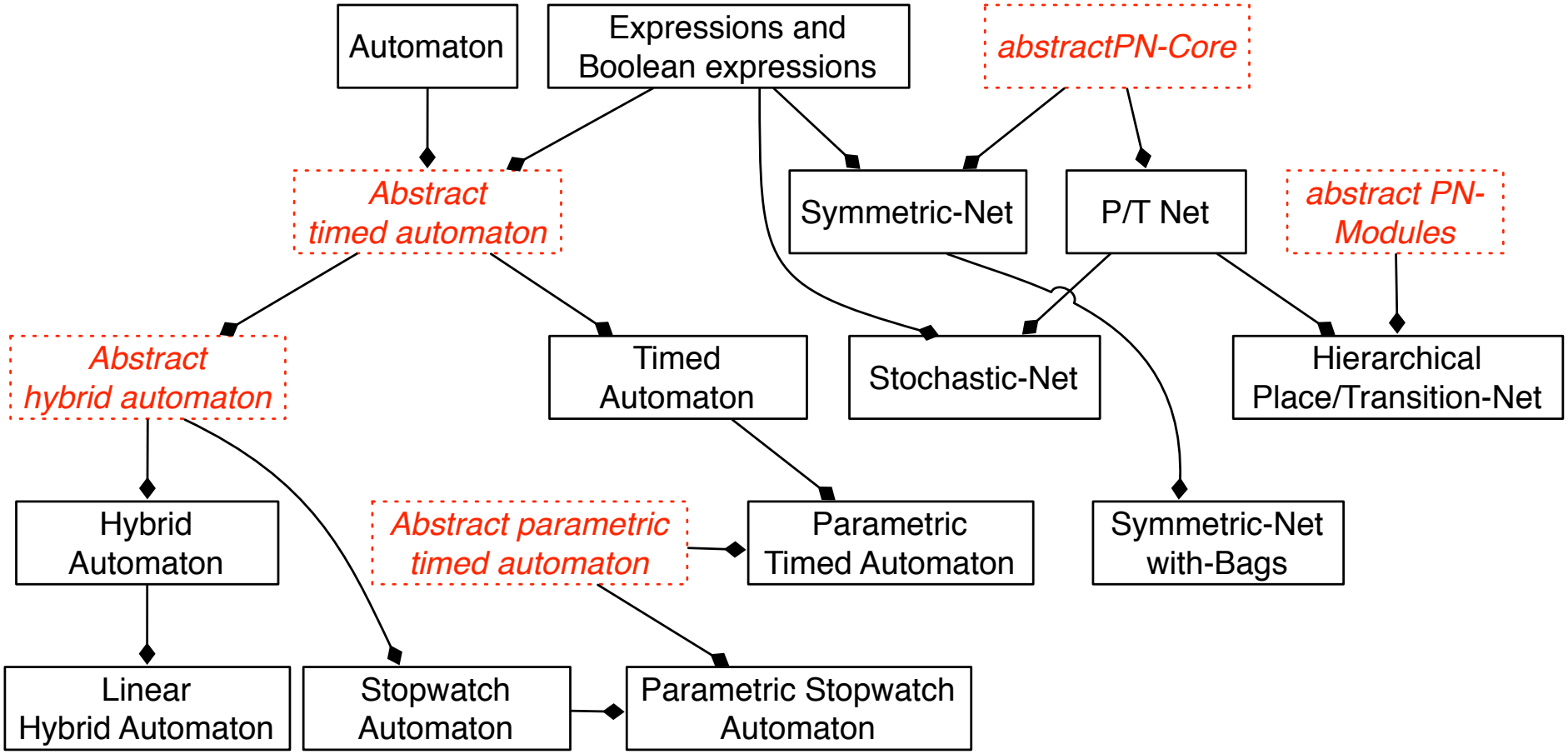
Example of a GrML code:

```
<?xml 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:



Application to Automata and Petri Nets



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

A Common and Reusable Syntax

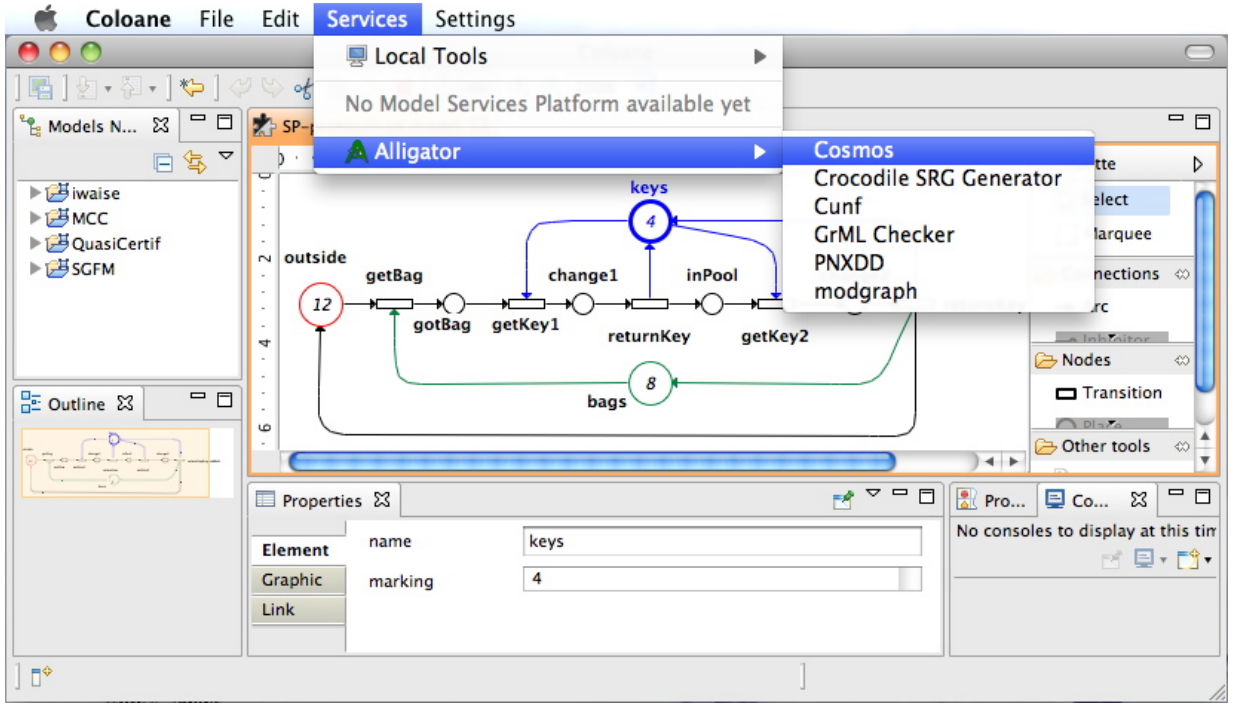
CosyVerif, a new integration platform [AHHKLLP13]

- 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

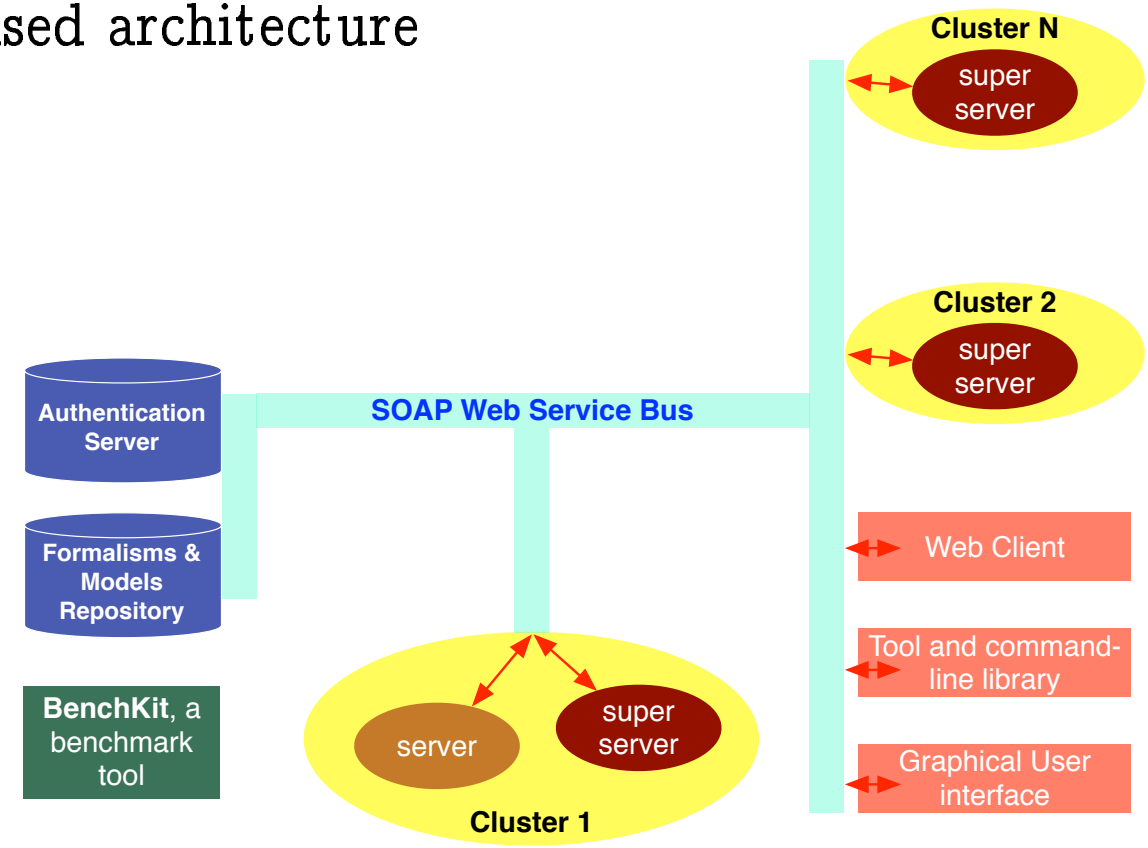
Web-services Client-Server Architecture

- Easy to install: simple and light multi-platform client connecting to servers



Web-services Client-Server Architecture

- Tool invocation through Web services transparent to the end-user
- Cloud-based architecture



Outline

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

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

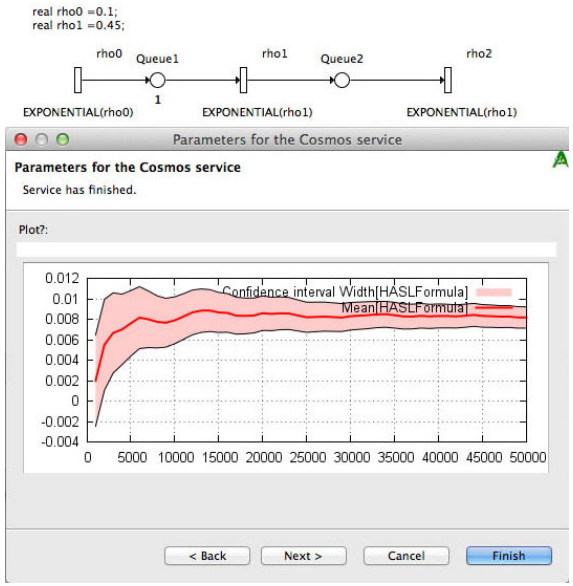
Two Bundles

Two bundles available (`http://download.cosyverif.org`)

All services Petri Nets

Statistical model checker [Ballarini 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]

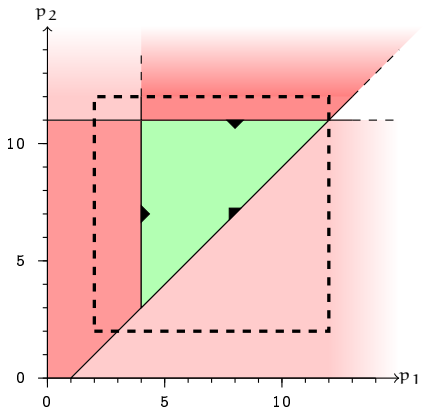
- Symbolic/symbolic approach based on Symmetric Nets with Bags [Haddad et al., 2009]
- Two symbolic techniques to counter state space explosion
  1. symmetries to reduce the reachability graph [Chiola et al., 1991]
  2. hierarchical Set Decision Diagrams to store the reachability graph [Couvreur and 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]

Outline

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

André, É., Fribourg, L., Kühne, U., and Soulat, R. (2012).
IMITATOR 2.5: A tool for analyzing robustness in scheduling problems.
In *Formal Methods*, volume 7436 of *Lecture Notes in Computer Science*, pages 33–36. Springer.

Baldan, P., Bruni, A., Corradini, A., König, B., Rodríguez, C., and Schwoon, S. (2012).
Efficient unfolding of contextual Petri nets.
Theoretical Computer Science, 449:2–22.

Ballarini, P., Djafri, H., Duflot, M., Haddad, S., and Pekergin, N. (2011).
HASL: An expressive language for statistical verification of stochastic models.
In *VALUETOOLS*, pages 306–315.

Chiola, G., Dutheillet, C., Franceschinis, G., and Haddad, S. (1991).
On well-formed coloured nets and their symbolic reachability graph.
In *ICATPN*. Springer-Verlag.

Colange, M., Baarir, S., Kordon, F., and Thierry-Mieg, Y. (2011).
Crocodile: A symbolic/symbolic tool for the analysis of symmetric nets with bags.
In *ICATPN*, volume 6709 of *Lecture Notes in Computer Science*, pages 338–347. Springer.

References II

Couvreur, J.-M. and Thierry-Mieg, Y. (2005).
Hierarchical decision diagrams to exploit model structure.
In *FORTE*, volume 3731 of *Lecture Notes in Computer Science*, pages 443–457. Springer.

Haddad, S., Kordon, F., Petrucci, L., Pradat-Peyre, J.-F., and Trèves, N. (2009).
Efficient state-based analysis by introducing bags in Petri net color domains.
In *ACC*, pages 5018–5025. Omnipress IEEE.

Hong, S., Kordon, F., Paviot-Adet, E., and Evangelista, S. (2012).
Computing a hierarchical static order for decision diagram-based representation from P/T nets.
Transactions on Petri Nets and Other Models of Concurrency, V:121–140.

Kordon, F., Linard, A., Becutti, M., Buchs, D., Fronc, L., Hulin-Hubard, F., Legond-Aubry, F., Lohmann, N., Marechal, A., Paviot-Adet, E., Pommereau, F., Rodrígues, C., Rohr, C., Thierry-Mieg, Y., Wimmel, H., and Wolf, K. (2013).
Web report on the model checking contest @ Petri Net 2013.
Available at `http://mcc.lip6.fr`.

References III

Kordon, F., Linard, A., Buchs, D., Colange, M., Evangelista, S., Lampka, K., Lohmann, N., Paviot-Adet, E., Thierry-Mieg, Y., and Wimmel, H. (2012).
Report on the model checking contest at Petri Nets 2011.
ToPNoC, V:121–140.

Lakos, C. and Petrucci, L. (2004).
Modular analysis of systems composed of semiautonomous subsystems.
In *ACSD*, pages 185–196. IEEE Computer Society.

Rodríguez, C. and Schwoon, S. (2012).
Verification of Petri nets with read arcs.
In *CONCUR*, volume 7454 of *Lecture Notes in Computer Science*, pages 471–485.

Rodríguez, C., Schwoon, S., and Baldan, P. (2011).
Efficient contextual unfolding.
In *CONCUR*, volume 6901 of *Lecture Notes in Computer Science*, pages 342–357.

Licensing

Source of the graphics used

Title: Scottish Kitten
Author: RN3DLL
Source: https://commons.wikimedia.org/wiki/File:Scottish_Kitten.png
License: CC BY-SA 3.0

License of this document

Author: The *CosyVerif* team