IOP : Intégration d'Outils à la Plate-forme *CosyVerif*

Étienne André, Laure Petrucci
Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, UMR 7030, F-93430, Villetaneuse, France
Fabrice Kordon
LIP6, CNRS UMR 7606, Université P. & M. Curie and Université Paris Ouest, France
Alban Linard
LSV, CNRS, INRIA & ENS Cachan, France

Abstract

CosyVerif aims at gathering within a common framework various existing tools for specification and verification. It has been designed in order to 1) support different formalisms with the ability to easily create new ones, 2) provide a graphical user interface for every formalism, 3) include verification tools called via the graphical interface or via an API as a Web service, and 4) offer the possibility for a developer to integrate his/her own tool without much effort, also allowing it to interact with the other tools. We present here a project that aims at integrating more tools into the *CosyVerif* platform.

1 Context

Formal verification of complex concurrent and heterogeneous systems often requires their model checking on complementary facets (such as discrete, timed, stochastic, etc.) of their behaviour. No single formalism being complete enough to encompass all these facets, such systems can consequently be modelled using different formalisms such as (different types of) Petri nets and timed automata. Various tools support these formalisms, each having different input and output syntaxes for models and analysis results. This often impedes integrated and comprehensive verification campaigns on complex concurrent and heterogeneous systems.

The IOP project aimed at integrating tools within the *CosyVerif* platform, a verification environment providing several formalisms and tools, and allowing for transparent tool invocations through Web services.

1.1 The *CosyVerif* Platform

CosyVerif [AHHH$^+$13] is a distributed and open verification environment that currently handles two families of formalisms: Petri nets and timed automata. So far, 12 declared concrete formalisms from these 2 families are available, interrelated through a modular architecture of definitions, reusing common concepts, and enabling easy addition of new notations. They are syntactically supported by a two-layered XML-based language: the Formalism Markup Language (FML, the superstructure) and the Graph Markup Language (GrML, the infrastructure) [ABD$^+$13].

Tools developers can declare a new formalism in the platform using FML, by reusing portions of existing formalisms (when they share common concepts). GrML is the internal representation of specifications in *CosyVerif*. FML and GrML ensure syntactic interoperability among tools that may only manipulate abstract syntax trees. These XML-based technologies enable rapid development and reuse of parsers and syntactic validation. Thanks to such facilities,

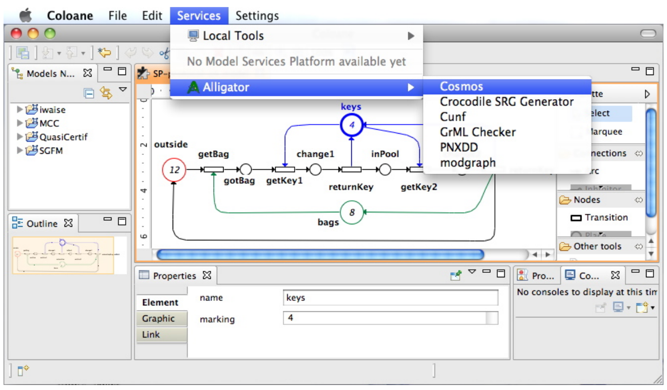

Figure 1 – Screenshot of the Coloane interface

the typical integration effort for tools developers is a day once they have been prepared (e.g. adaptation of inputs and outputs). More work is required for pre-existing tools that are ported to populate the new verification environment (e.g. several tools from the CPN-AMI Petri net verification environment that is being replaced by *CosyVerif*).

CosyVerif is an open distributed environment that can be enriched by any researcher willing to contribute. A registration mechanism allows for the diffusion of any services over a federation of *CosyVerif* nodes, which greatly improves the time-to-availability for new tools.

Tools are invoked through Web services transparently to end users, thanks to Coloane, an open source extensible graphical editor based on Eclipse (see Fig. 1). It offers modelling facilities and a way to apply tools services on models. Since *CosyVerif* relies on Web services, the use of Coloane is not mandatory and verification services can be accessed directly via the underlying XML-based protocol.

The *CosyVerif* project also provides a repository of models, that may be used for benchmark purposes. These models mostly come from industrial real-time case studies and the Model Checking Contest in 2011 [KLB+12a] and 2012 [KLB+12b].

1.2 Verification Tools in *CosyVerif*

Before this project, the following tools were integrated into *CosyVerif*:

- Cosmos [BDD+11]: a statistical model checker;
- Crocodile [CBKTM11]: tool for the so-called symbolic/symbolic approach dealing with Symmetric Nets with Bags [HKP+09];

  item CUNF [BBC+12]: toolset for carrying out *unfolding-based* verification of Petri nets extended with read arcs, also called *contextual nets* (c-nets);
- IMITATOR [AFKS12]: tool for the parameter synthesis for parametric timed automata augmented with variables and stopwatches;
- LoLA [Wol07]: explicit Petri Net state space verification tool;
- PNXDD [HKPAE12]: tool generating the state space and evaluating CTL formulæ on P/T nets.

2 Objectives of the Project and Results

The main objective of this action is to integrate more tools into the *CosyVerif* platform. In particular, some of these tools were extracted from predecessors of this environment (in particular, CPN-AMI).

A second objective was to establish an integration procedure that would benefit for other tools to be integrated in *CosyVerif*.

This has been done thanks to two interns hired at LIPN and LIP6, viz., Henoc Khouilla and Idrissa Sokhona.

In both cases, the integration procedure was carefully thought. The already available procedure has been documented by the interns, and adjusted to meet their specific requirements.

We nosw list the tool that were integrated in *CosyVerif*.

GreatSPN invariant computation for Petri nets GreatSPN is a well know Petri net tool that offers numerous services (model checking, stochastic analysis and invariant computation). CPN-AMI integrated the invariant computation modules from its user interface. This module computes:

- Place invariants,
- Transition invariants,
- Minimal syphon,
- Minimal traps.

We hired a student that prepared the integration of these functions in *CosyVerif* by translating the internal *CosyVerif* format into the one of CPN-AMI, thus enabling the reuse of the previous translators. Then, testing and benchmarking was done to access the new integration's results compared to ones provided by CPN-AMI.

Petri net structural bound computation CPN-AMI offered a tool to compute structural bounds of a tool based on the net's structure (i.e. with a lower complexity than the precise bounds to be computed by model checking). Based on the previous experience of the greatSPN integration, a similar work was done for this service, that could benefit from part of the previous work.

ModGraph [LP04] This tool performs construction and analysis of modular state spaces. Instead of actually synchronising a set of automata sharing some common transitions, it builds a synchronisation structure and keeps only the reachable parts of the automata. Thus, interleaving is avoided as much as possible. The tool also provides some analysis features, in particular reachability and deadlock-checking, that can be specified only on a subset of the interacting modules.

This tool had previously been integrated in *CosyVerif*, but it provided only a poor user interface. For instance, all the results were in a big text field, whereas *CosyVerif* services should show them as several typed fields.

The internship was in two parts:

- upgrade the ModGraph service to the latest version of the tool;
- enhance the user interface provided by the service.

ObsGraph [KO12] This BDD-based tool implements a verification approach for workflows using Symbolic Observation Graphs [HIK04]. This approach abstracts the given workflows, described as Petri net models, allowing for confidentiality (e.g. to preserve companies internal processes), and showing only the actions meant to be composed with other actions. Deadlock verification is therefore reduced to verifying only the synchronised product of the abstractions corresponding to the components.

As for ModGraph, this tool had previously been integrated in *CosyVerif*, but it provided only a poor user interface. Again, all the results were in a big text field, whereas *CosyVerif* services should show them as several typed fields.

The internship was in three parts:

- upgrade the ModGraph service to the latest version of the tool;
- enhance the user interface provided by the service;
- upgrade the service by interaction with the tool developer, for instance the addition of new services above the ObsGraph tool.

Helena [hel] Helena is an explicit state model checker. A High-level Petri net is used for models. Helena features an efficient firing rule mechanism and code generation to speed up the analysis. It provides an interface with C code. It also implements different techniques for efficient state space analysis: optimised state space storage, partial order methods and allows for LTL model checking.

The internship was a first attempt to the integration of Helena in *CosyVerif*. A prototype was obtained, but not polished enough to be released yet. Integrating Helena is difficult because a translation from the *CosyVerif* model format to Helena's one must be defined.

3 Perspectives

Perspectives for the *CosyVerif* platform include the integration of new tools, but also several improvements on the server side.

Asynchronous Tool Invocation The end user will be able to launch a verification process, and get the result later, even if the connection between the graphical client (e.g. Coloane) and the server is broken. In future releases, the result could also be for instance sent by email when the verification is finished.

Command-Line Client Tools in *CosyVerif* are not intended to be accessed only via the provided user interface. If this can be useful for demonstration or educational purposes, direct access via web services is also of interest. For instance, *CosyVerif* could be used as a back-end verification platform for other tools dedicated to higher order languages like AADL or VHDL via a transformation into one of the available formalisms. To ease the integration of such tools, a basic command-line library is being developed.

Federation of Servers In order to ease deployment and perform load balancing over a set of servers, *CosyVerif* will integrate the transparent construction of a federation of servers. The user still connects to his/her usual server that also acts as a proxy for the whole federation. Then, services are executed on the less loaded machine among those that provide it.

Acknowledgements

The development of the *CosyVerif* integration platform was supported by: LIP6, the FEDER Île-de-France/System@tic–free software thanks to the NEOPPOD project (support of two engineers), LIPN (support of one engineer), as well as LSV and Inria (support of several engineers). We finally would like to warmly thank GDR GPL and in particular Laurence Duchien for the support to the *CosyVerif* platform.

References

[ABD+13] Étienne André, Benoît Barbot, Clément Démoulins, Lom Messan Hillah, Francis Hulin-Hubard, Fabrice Kordon, Alban Linard, and Laure Petrucci. A modular approach for reusing formalisms in verification tools of concurrent systems. In Lindsay Groves and Jing Sun, editors, *15th International Conference on Formal Engineering Methods (ICFEM'13)*, volume 8144 of *Lecture Notes in Computer Science*, pages 199–214. Springer, October 2013.

[AFKS12] Étienne André, Laurent Fribourg, Ulrich Kühne, and Romain Soulat. 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, 2012.

[AHHH+13] Étienne André, Lom-Messan Hillah, Francis Hulin-Hubard, Fabrice Kordon, Yousra Lembachar, Alban Linard, and Laure Petrucci. CosyVerif: An open source extensible verification environment. In Yang Liu and Andrew Martin, editors, *18th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'13)*, pages 33–36. IEEE Computer Society, July 2013.

[BBC+12] P. Baldan, A. Bruni, A. Corradini, B. König, C. Rodríguez, and S. Schwoon. Efficient unfolding of contextual Petri nets. *Theoretical Computer Science*, 449:2–22, 2012.

[BDD+11] Paolo Ballarini, Hilal Djafri, Marie Duflot, Serge Haddad, and Nihal Pekergin. HASL: An expressive language for statistical verification of stochastic models. In *VALUETOOLS*, pages 306–315, 2011.

[CBKTM11] M. Colange, S. Baarir, F. Kordon, and Y. Thierry-Mieg. 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, 2011.

[hel] http://lipn.univ-paris13.fr/~evangelista/helena/.

[HIK04] Serge Haddad, Jean-Michel Ilié, and Kais Klai. Design and evaluation of a symbolic and abstraction-based model checker. In *ATVA*, pages 196–210, 2004.

[HKP+09] S. Haddad, F. Kordon, L. Petrucci, J-F. Pradat-Peyre, and N. Trèves. Efficient state-based analysis by introducing bags in Petri net color domains. In *ACC*, pages 5018–5025. Omnipress IEEE, 2009.

[HKPAE12] S. Hong, F. Kordon, E. Paviot-Adet, and S. Evangelista. 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, 2012.

[KLB+12a] F. Kordon, A. Linard, D. Buchs, M. Colange, S. Evangelista, K. Lampka, N. Lohmann, E. Paviot-Adet, Y. Thierry-Mieg, and H. Wimmel. Report on the model checking contest at Petri nets 2011. *Transactions on Petri Nets and Other Models of Concurrency*, VI:169–196, 2012.

[KLB+12b] Fabrice Kordon, Alban Linard, Didier Buchs, Maximilien Colange, Sami Evangelista, Lukasz Fronc, Lom-Messan Hillah, N. Lohmann, Emmanuel Paviot-Adet, Franck Pommereau, C. Rohr, Yann Thierry-Mieg, Harro Wimmel, and Karsten Wolf. Raw report on the model checking contest at Petri nets 2012. Technical report, 2012. CoRR.

[KO12] Kais Klai and Hanen Ochi. Modular verification of inter-enterprise business processes. In *eKNOW*, pages 155–161, 2012.

[LP04] Charles Lakos and Laure Petrucci. Modular analysis of systems composed of semiautonomous subsystems. In *ACSD*, pages 185–196. IEEE Computer Society, 2004.

[Wol07] Karsten Wolf. Generating Petri net state spaces. In *ICATPN*, volume 4546 of *Lecture Notes in Computer Science*, pages 29–42. Springer, 2007.