http://www-lipn.univ-paris13.fr/~petrucci/PAPERS
In Proc. of the 18th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'13), Singapore, pp. 33–36, IEEE Computer Society Press, July 2013.

CosyVerif: an Open Source Extensible Verification Environment

Étienne André, Yousra Lembachar, Laure Petrucci
Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, UMR 7030, F-93430, Villetaneuse, France
Francis Hulin-Hubard, Alban Linard
LSV, CNRS, INRIA & ENS Cachan, France
Lom Hillah, Fabrice Kordon
LIP6, CNRS UMR 7606, Université P. & M. Curie and Université Paris Ouest, 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. Several tools have already been integrated for the formal verification of (extensions of) Petri nets and timed automata.**

Keywords*—*Formal verification, Distributed computing, Client-server systems, Web services, Software reusability, Software architecture.

I. Introduction

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.

This paper presents *CosyVerif*, a verification environment that integrates several formalisms and tools, and allows for transparent tool invocations through Web services.

Section II describes the characteristics of the *CosyVerif* platform. Then Section III briefly presents the currently integrated tools, before Section IV sketches ongoing work prior to a comparison with similar existing environments in Section V.

II. The *CosyVerif* Environment

CosyVerif [1] 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).

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, the typical integration effort for tools developers is half a day.

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 [2] and 2012 [3].

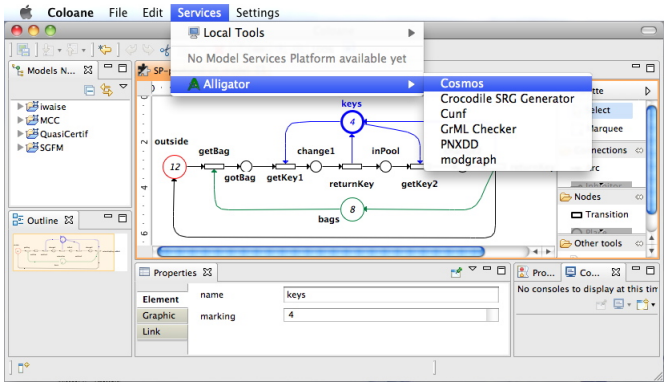

Fig. 1: Screenshot of the Coloane interface

III. VERIFICATION TOOLS IN *CosyVerif*

Let us present some tools currently integrated in *CosyVerif*.

a) Cosmos [4]: This statistical model checker takes as input Generalised Stochastic Petri Nets with general distribution (GSPN) and a Hybrid Automaton Stochastic Logic (HASL) formula, and returns the statistical estimation of the formula with a confidence interval (see Fig. 2). HASL allows for selecting some trajectories of the model and thus allows model checking. HASL can also be used to define complex performance indexes on the model. It has recently been extended to support rare event acceleration using importance sampling techniques.

b) Crocodile [5]: This tool for the so-called symbolic/symbolic approach deals with Symmetric Nets with Bags [6]. It combines two techniques for handling the combinatorial explosion of the state space that are both called symbolic. The first symbolic technique concerns the reduction of the reachability graph of a system using its symmetries [7]. The second symbolic technique consists in storing the reachability graph using decision diagrams. The SDD (Set Decision Diagrams [8]), the class of decision diagrams we use, are hierarchical. Thus we can exploit this aspect when storing in a very compact way the state space of such systems [5]. Crocodile generates the state space and evaluates CTL formulæ.

c) CUNF *[9]:* This is a toolset for carrying out *unfolding-based* verification of Petri nets extended with read arcs, also called *contextual nets* (c-nets). Unfoldings fully represent the state-space (reachable markings) of a c-net by a partial order rather than by a set of interleavings; they are often exponentially smaller than the reachability graph, and never larger. Additionally, c-net unfoldings can be exponentially more compact than those of corresponding Petri nets, see [9]. The toolset contains in particular an unfolding construction tool [10] and a reachability and deadlock checking tool [11].

d) IMITATOR *[12]:* This tool performs parameter synthesis for parametric timed automata [13] augmented with variables and stopwatches. In particular, it implements the

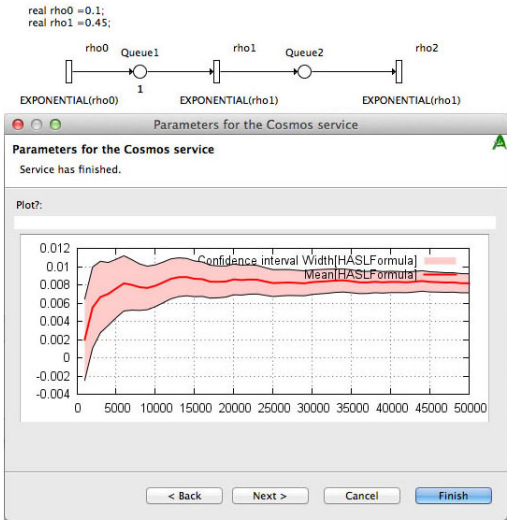

Fig. 2: Typical output of Cosmos

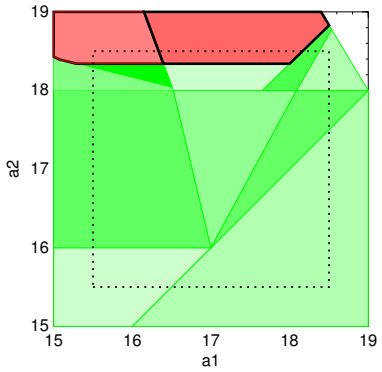

Fig. 3: Cartography output by IMITATOR

"inverse method" [14]: from a single valuation of the parameters, it computes a convex constraint around this reference valuation guaranteeing the same (time-abstract) behaviour. A major advantage is to give a quantitative measure of the system robustness. IMITATOR also implements the "behavioural cartography" [14]: this algorithm covers a subset of the parametric space with *tiles*, i.e. parametric zones where the system behaviour is uniform. An example of cartography as output by IMITATOR is given in Fig. 3.

e) LoLA [15]: This is an explicit Petri Net state space verification tool. It can verify a variety of properties ranging from questions regarding individual nodes (e.g. boundedness of a place or quasi-liveness of a transition), reachability of a given state or a state predicate, typical questions related to a net (e.g. deadlock freedom, reversibility, or boundedness), and the validity of temporal logical formulæ such as CTL. It has been successfully used in case studies from various domains, including asynchronous circuits, biochemical reaction chains, services, business processes, and parameterised Boolean programs. For each property, LoLA provides tailored versions of state space reduction techniques such as stubborn sets, symmetry reduction, coverability graph generation, or methods involving the Petri Net invariant calculus. Depending on the property to be preserved, these techniques can also be used in combination to only generate a small portion of the state space.

This tool has been developed at the University of Rostock and we only integrated it in *CosyVerif* (requiring us to write a translator between GrML and LoLA's internal format).

f) ModGraph [16]: 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.

g) ObsGraph [17]: This BDD-based tool implements a verification approach for workflows using Symbolic Observation Graphs [18]. 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.

h) PNXDD [19]: This tool generates the state space and evaluates CTL formulæ on P/T nets. It also handles Symmetric nets through their unfolding into an equivalent P/T net. PNXDD exploits hierarchy: a state is seen as a tree, where the leaves correspond to place markings. This particular structure offers better sharing opportunities than, for instance, a vector-based representation. The conception of such a tree is critical to reach good performances and heuristics are being elaborated for this purpose. As for Crocodile, PNXDD relies on SDD [8].

IV. Evolutions of the *CosyVerif* Environment

CosyVerif is a long term project intended for numerous evolutions. We present here the ones we are currently working on. They should be available soon.

1) Asynchronous Tool Invocation: The end user can 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.

2) 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 education purposes, direct access via web services is also of interest. For instance, *CosyVerif* can 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.

3) 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 least loaded machine among those that provide it.

Fig. 4 presents a typical architecture of such a federation. Servers can be grouped into clusters containing at least one "super server" that both serves as a proxy to clients (graphical user interface, command-line, etc.) and communicate together

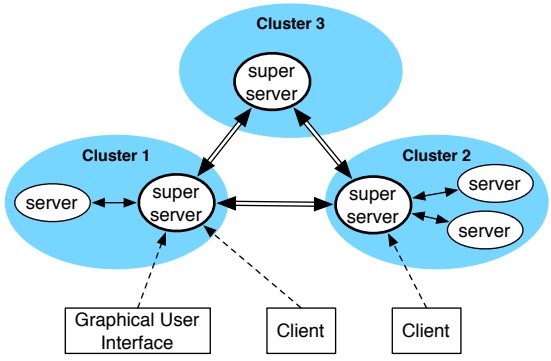

Fig. 4: Typical architecture of a federation of *CosyVerif* servers

their respective list of services. When a client contacts its corresponding super server, it gets the list of services provided by the whole federation. This mechanism is of course fully transparent to the end user. The default configuration is a server cluster, then acting as a super server.

4) Enhanced Interaction Between Tools: Since tools all rely on unified formalisms, one can easily use the output of a tool as an input of another tool. This will allow for automated or user-defined tool chaining and, hence enhance their potential complementarity.

5) Common syntax for the model editors: Modellers can use a consistent concrete syntax for expressions and formulæ, whatever the formalism used. The *CosyVerif* environment is thus easier to learn and use.

More powerful interactions could take place using semantical aspects. So far, tools only agree on a unified input abstract syntax; adding semantics will allow for an automatic mapping of common concepts between two formalisms. A rather short-term future work is to handle these aspects inside the family of formalisms of timed automata on the one hand, and of Petri nets on the other hand. For Petri nets, the ongoing standardisation [20] already provides some guidelines. A longer term future work is be to build connections between the two families, as well as with other families of formalisms.

V. Related Platforms

Several platforms have been designed over the past decade in order to achieve similar goals. CASL (Common Algebraic Specification Language) is a general-purpose specification language. A tool named HetCASL[1] (Heterogeneous Tool Set) has been proposed, that incorporates different theorem provers and different specification languages, hence allowing the designer to handle heterogeneous specifications. This approach is very much theorem prover oriented (including connections with Isabelle, Maude, etc.). In contrast, *CosyVerif* is more general.

Diabelli [21] is a heterogeneous proof system, allowing one to perform theorem proving with both diagrammatic and sentential formulæ, and proof steps. It is shipped as a standalone tool combining Isabelle and Speedith. The tool does feature a graphical interface, but models are given in a textual form only. We believe that this tool does not provide a high degree of flexibility (because it requires translations), and apparently it does not work in the cloud, contrarily to *CosyVerif*.

LTSmin [22] is a meta toolkit that supports different input language modules (mCRL2, Promela, etc.) relying on labelled transition systems (LTS). LTSmin allows LTS-based semantic exchanges of state space between different tools (based on a Partitioned Next-State function). Furthermore, it allows the end user to apply alternative verification algorithms to their native tool. This is of high interest. However, the tool only works with a LTS-based semantics, whereas we aim at considering a larger set of formalisms.

Rich-model Toolkit[2] is a standardisation of formal languages: it features common formats for systems, formulæ,

[1] http://www.informatik.uni-bremen.de/agbkb/forschung/formal_methods/CoFI/hets/

[2] http://richmodels.epfl.ch/

proofs and counterexamples. Contrarily to our approach, it is SAT- and SMT-oriented, and algorithms seem to be built-in, although it is hard to get a precise idea of the features, since this is a very recent initiative.

StarExec[3] is an initiative of the logic community to build a shared logic solving infrastructure (SAT, SMT), to enable researchers to manage libraries, provide solver execution on a large cluster, and facilitate translation between logics. According to their system architecture specification, users interface with the infrastructure via a Web application.

PAT [23] is a multi-formalisms platform based on modules. Each module relies on its own formalism and domain of application (e.g. real-time systems [24], probabilistic systems, network calculus, etc.), and must provide a semantics in the form of LTS. Then, common algorithms (deadlock-checking, LTL-checking) can be used for any of the modules, in addition to domain-specific algorithms. It also features graphical facilities, a simulator, syntactical checkers, counterexample exhibition, etc. Different from our approach, PAT is mainly LTS-based (with additional integration of Markov Decision Processes and Timed Transition Systems), and formalisms are not related to each other, i.e. the modules are independent.

ACKNOWLEDGEMENT

The development of the *CosyVerif* integration platform was supported by: LIP6 and 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).

REFERENCES

[1] The CosyVerif group, "CosyVerif home page," http://cosyverif.org.

[2] 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*, vol. VI, pp. 169–196, 2012.

[3] F. Kordon, A. Linard, D. Buchs, M. Colange, S. Evangelista, L. Fronc, L.-M. Hillah, N. Lohmann, E. Paviot-Adet, F. Pommereau, C. Rohr, Y. Thierry-Mieg, H. Wimmel, and K. Wolf, "Raw report on the model checking contest at Petri nets 2012," Tech. Rep., 2012, coRR.

[4] P. Ballarini, H. Djafri, M. Duflot, S. Haddad, and N. Pekergin, "HASL: An expressive language for statistical verification of stochastic models," in *VALUETOOLS*, 2011, pp. 306–315.

[5] 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*, ser. Lecture Notes in Computer Science, vol. 6709. Springer, 2011, pp. 338–347.

[6] 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*. Omnipress IEEE, 2009, pp. 5018–5025.

[7] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad, "On well-formed coloured nets and their symbolic reachability graph," in *ICATPN*. Springer-Verlag, 1991.

[8] J.-M. Couvreur and Y. Thierry-Mieg, "Hierarchical decision diagrams to exploit model structure," in *FORTE*, ser. Lecture Notes in Computer Science, vol. 3731. Springer, 2005, pp. 443–457.

[9] P. Baldan, A. Bruni, A. Corradini, B. König, C. Rodríguez, and S. Schwoon, "Efficient unfolding of contextual Petri nets," *Theoretical Computer Science*, vol. 449, pp. 2–22, 2012.

[3] http://www.starexec.org/starexec/public/about.jsp

[10] C. Rodríguez, S. Schwoon, and P. Baldan, "Efficient contextual unfolding," in *CONCUR*, ser. Lecture Notes in Computer Science, vol. 6901, 2011, pp. 342–357.

[11] C. Rodríguez and S. Schwoon, "Verification of Petri nets with read arcs," in *CONCUR*, ser. Lecture Notes in Computer Science, vol. 7454, 2012, pp. 471–485.

[12] É. André, L. Fribourg, U. Kühne, and R. Soulat, "IMITATOR 2.5: A tool for analyzing robustness in scheduling problems," in *Formal Methods*, ser. Lecture Notes in Computer Science, vol. 7436. Springer, 2012, pp. 33–36.

[13] R. Alur, T. A. Henzinger, and M. Y. Vardi, "Parametric real-time reasoning," in *STOC*. ACM, 1993, pp. 592–601.

[14] É. André and R. Soulat, *The Inverse Method*. ISTE Ltd and John Wiley & Sons Inc., 2013.

[15] K. Wolf, "Generating Petri net state spaces," in *ICATPN*, ser. Lecture Notes in Computer Science, vol. 4546. Springer, 2007, pp. 29–42.

[16] C. Lakos and L. Petrucci, "Modular analysis of systems composed of semiautonomous subsystems," in *ACSD*. IEEE Computer Society, 2004, pp. 185–196.

[17] K. Klai and H. Ochi, "Modular verification of inter-enterprise business processes," in *eKNOW*, 2012, pp. 155–161.

[18] S. Haddad, J.-M. Ilié, and K. Klai, "Design and evaluation of a symbolic and abstraction-based model checker," in *ATVA*, 2004, pp. 196–210.

[19] 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*, vol. V, pp. 121–140, 2012.

[20] L. Hillah, F. Kordon, C. Lakos, and L. Petrucci, "Extending PNML scope: A framework to combine Petri nets types," *Transactions on Petri Nets and Other Models of Concurrency*, vol. VI, pp. 46–70, 2012.

[21] M. Urbas and M. Jamnik, "Diabelli: A heterogeneous proof system," in *IJCAR*, ser. Lecture Notes in Computer Science, vol. 7364. Springer, 2012, pp. 559–566.

[22] S. Blom, J. van de Pol, and M. Weber, "LTSmin: Distributed and symbolic reachability," in *CAV*, ser. Lecture Notes in Computer Science, vol. 6174. Springer, 2010, pp. 354–359.

[23] Y. Liu, J. Sun, and J. S. Dong, "PAT 3: An extensible architecture for building multi-domain model checkers," in *ISSRE*. IEEE, 2011, pp. 190–199.

[24] J. Sun, Y. Liu, J. S. Dong, Y. Liu, L. Shi, and É. André, "Modeling and verifying hierarchical real-time systems using Stateful Timed CSP," *ACM Transactions on Software Engineering and Methodology*, vol. 22, no. 1, pp. 3.1–3.29, 2013.