Engineer position

Building a new input syntax for the IMITATOR parametric timed model checker

PI: Étienne André
Email: engineer@loria.science
Laboratory: LORIA, Université de Lorraine, CNRS, Inria, Nancy, France
Team: MOSEL + VeriDis

1 Context: IMITATOR

IMITATOR [And+12] is a *parametric timed model checker*, performing parametric verification and robustness analysis of models of systems featuring concurrency, hard timing constraints, and uncertain or unknown timing constraints. That is, IMITATOR can answer questions such as:

1. what are the admissible periods such that a real-time system meets its deadlines?
2. by how much can we vary some timing constants of a system such that the system is robust, *i.e.* its discrete behavior remains identical?
3. or, more pragmatically (but not less importantly), what are the `Wait` values for which a coffee machine guarantees users to deliver a coffee within 10 seconds?

IMITATOR takes as input formalism the well-known formalism of *parametric timed automata* [AHV93], a useful but highly complex extension of finite-state automata with clocks [AD94] and unknown constants (*i.e.* parameters). The formalism supported by IMITATOR extends parametric timed automata with integer variables and stopwatches. Parameters can be used both in the model and in the properties. Verification capabilities include reachability-synthesis, a subset of the TCTL logics, deadlock-freeness-synthesis, non-Zeno model checking, and trace-preservation-synthesis. IMITATOR is fully written in OCaml, and makes use of the Parma Polyhedra Library [BHZ08]. It is available under the GNU General Public License. It also features distributed capabilities to run over a cluster [ACN15] and, while it features no graphical user interface, it is able to output graphics (see *e.g.* Fig. 1).

2 Public and applications

Target The targeted users are not only academic with an expertise in formal methods, but also industrial practionners with less expertise in formal topics.

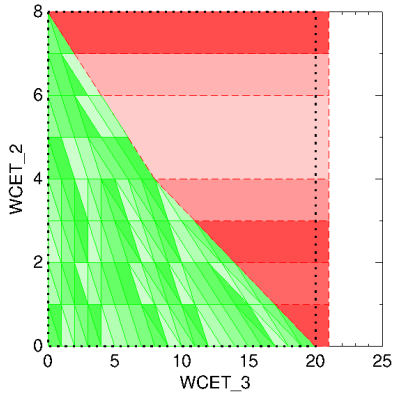

Figure 1: Example of graphical output

Applications IMITATOR not only has led to publications in top-end venues (such as TACAS [And+19a], CAV [WAH19], and a best paper award at the core rank A conference ICECCS [AHW18]), including by other research groups [Aşt+16; Lut+19], but is also used in several applicative contexts.

IMITATOR was used in various research groups related, among other areas, to parametric schedulability analysis (*e.g.* [SAL15; AJM19]), product lines testing [Lut+19], monitoring cyber-physical systems [AHW18], and even analysis of music scores [FJ13]. Recently, we also used IMITATOR to guarantee cybersecurity properties [AS19].

Industrial collaborations In addition to these academic works, IMITATOR was used and is still used in several industrial collaborations:

- verification of an asynchronous memory circuit with ST-microelectronics [And+12];
- scheduling analysis with Astrium Space Transportation [Fri+12];
- schedulability analysis with Thales [SAL15; Par+16], and supporting Thales' "Time4sys" formalism [And19; AJM19];
- verification of an industrial asynchronous leader election algorithm by Thales [And+19c];
- parametric schedulability for ArianeGroup [And+19b].

3 Main objective

Despite some usage in the industry, IMITATOR clearly lacks a lot of handufl features for a model checker. This certainly hinders the usage of the tool by industrial practicionners lacking an expertise in formal methods.

This leads companies to regularly write feature requests, that the current developers have very little towards no time to develop.

Main goal The main goal is **to improve the IMITATOR input language**. For example, the only "data structure" currently supported is shared unbounded exact rationals. To encode an array, one should create as many variables (of type unbounded exact rational) as the size of the array, plus one variable to encode the number of elements in the array. This makes the models truely unreadable and unnecessarily complicated.

Therefore, the main goal will be to significantly improve the input syntax, so as to ease the use and the dissemination of the tool. This may include:

1. allowing new single types: Booleans, small integers, large integers, binary words (requested by an industrial user), with the associated operations;
2. allowing new composite types: enumerate types, structured types, arrays, lists, etc., with the associated operations;
3. allowing user-defined types and functions;
4. allowing the definition and composition of an unbounded (but fixed) number of components: extremely useful to model a protocol with n clients (n fixed but large);
5. allowing to partially reuse models, with variables renaming.

4 Framework and conditions

4.1 Conditions

Requirements A degree of engineer in computer science (or an equivalent degree) is required. A PhD is *not* required.

Skills One or more of the following skills would be appreciated, though not compulsory: OCaml programming, Python programming, formal methods, model checking.

Enjoying working in a dynamic scientific environment will be appreciated.

The knowledge of French language is *not* compulsory for the position.

Conditions The successful applicant will be hired by Inria for 24 months.

Tentative starting date: 1st January 2021.

Competitive salary, including health insurance, and subsidies for public transportation and canteen (lunch).

Application Interested applicants are advised to get in touch with Étienne André first.
Dedicated contact email: `engineer@loria.science`

4.2 Location: Nancy

The position will take place at LORIA (Laboratoire lorrain de recherche en informatique et ses applications) at Université de Lorraine, Nancy. LORIA is an internationally recognized research laboratory comprising over 400 scientists from 48 nationalities.

Université de Lorraine is a dynamic university in the beautiful city of Nancy, 1h30 from Paris by TGV (high-speed train); Nancy is a human-sized city featuring a high quality of life, a UNESCO-world-heritage city center, and very affordable living costs.

Figure 2: Place Stanislas (author: Nicolas Cornet, license CC-by-sa)

5 Keywords

Tool development, software engineering, model checking, formal methods, parameter synthesis, parametric verification

References

[ACN15] Étienne André, Camille Coti, and Hoang Gia Nguyen. "Enhanced Distributed Behavioral Cartography of Parametric Timed Automata". In: *ICFEM*. (Nov. 3–6, 2015). Ed. by Michael Butler, Sylvain Conchon, and Fatiha Zaïdi. Vol. 9407. Lecture Notes in Computer Science. Paris, France: Springer, Nov. 2015, pp. 319–335. ISBN: 978-3-319-25422-7. DOI: 10.1007/978-3-319-25423-4_21.

[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)90010-8.

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. “Parametric real-time reasoning”. In: *STOC*. (May 16–18, 1993). Ed. by S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal. San Diego, California, United States: ACM, 1993, pp. 592–601. ISBN: 0-89791-591-7. DOI: 10.1145/167088.167242.

[AHW18] Étienne André, Ichiro Hasuo, and Masaki Waga. “Offline timed pattern matching under uncertainty”. In: *ICECCS*. (Dec. 12–14, 2018). Ed. by Anthony Widjaja Lin and Jun Sun. Melbourne, Australia: IEEE Computer Society, 2018, pp. 10–20. DOI: 10.1109/ICECCS2018.2018.00010.

[AJM19] Étienne André, Jawher Jerray, and Sahar Mhiri. “Time4sys2imi: A tool to formalize real-time system models under uncertainty”. In: *ICTAC*. (Oct. 31–Nov. 4, 2019). Ed. by Robert M. Hierons and Mohamed Mosbah. Vol. 11884. Hammamet, Tunisia: Springer, 2019, pp. 113–123. DOI: 10.1007/978-3-030-32505-3_7.

[And+12] Étienne André, Laurent Fribourg, Ulrich Kühne, and Romain Soulat. “IMITATOR 2.5: A Tool for Analyzing Robustness in Scheduling Problems”. In: *FM*. (Aug. 27–31, 2012). Ed. by Dimitra Giannakopoulou and Dominique Méry. Vol. 7436. Lecture Notes in Computer Science. Paris, France: Springer, Aug. 2012, pp. 33–36. DOI: 10.1007/978-3-642-32759-9_6.

[And+19a] Étienne André, Vincent Bloemen, Laure Petrucci, and Jaco van de Pol. “Minimal-Time Synthesis for Parametric Timed Automata”. In: *TACAS, Part II*. (Apr. 8–11, 2019). Ed. by Tomáš Vojnar and Lijun Zhang. Vol. 11428. Lecture Notes in Computer Science. Prague, Czech Repubic: Springer, 2019, pp. 211–228. DOI: 10.1007/978-3-030-17465-1_12.

[And+19b] Étienne André, Emmanuel Coquard, Laurent Fribourg, Jawher Jerray, and David Lesens. “Scheduling synthesis for a launcher flight control using parametric stopwatch automata”. In: *ACSD*. (June 27, 2019). Ed. by Jörg Keller and Wojciech Penczek. Aachen, Germany: IEEE, 2019, pp. 13–22. DOI: 10.1109/ACSD.2019.00006.

[And+19c] Étienne André, Laurent Fribourg, Jean-Marc Mota, and Romain Soulat. “Verification of an industrial asynchronous leader election algorithm using abstractions and parametric model checking”. In: *VMCAI*. (Jan. 13–15, 2019). Ed. by Constantin Enea and Ruzica Piskac. Vol. 11388. Lecture Notes in Computer Science. Lisbon, Portugal: Springer, 2019, pp. 409–424. DOI: 10.1007/978-3-030-11245-5_19.

[And19] Étienne André. “Formalizing Time4sys using parametric timed automata”. In: *TASE*. (July 29–Aug. 1, 2019). Ed. by Dominique Méry and Shengchao Qin. Guilin, China: IEEE, 2019, pp. 176–183. DOI: 10.1109/TASE.2019.00031.

[AS19] Étienne André and Jun Sun. “Parametric Timed Model Checking for Guaranteeing Timed Opacity”. In: *ATVA*. (Oct. 28–31, 2019). Ed. by Yu-Fang Chen, Chih-Hong Cheng, and Javier Esparza. Vol. 11781. Lecture Notes in Computer Science. Taipei, Taiwan: Springer, 2019, pp. 115–130. DOI: 10.1007/978-3-030-31784-3_7.

[Aşt+16] Lăcrămioara Aştefănoaei, Saddek Bensalem, Marius Bozga, Chih-Hong Cheng, and Harald Ruess. “Compositional Parameter Synthesis”. In: *FM*. Ed. by John S. Fitzgerald, Constance L. Heitmeyer, Stefania Gnesi, and Anna Philippou. Vol. 9995. Lecture Notes in Computer Science. 2016, pp. 60–68. DOI: 10.1007/978-3-319-48989-6_4.

[BHZ08] Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. “The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems”. In: *Science of Computer Programming* 72.1–2 (2008), pp. 3–21. DOI: 10.1016/j.scico.2007.08.001.

[FJ13] Léa Fanchon and Florent Jacquemard. “Formal Timing Analysis Of Mixed Music Scores”. In: *ICMC*. (Aug. 12–16, 2013). Perth, Australia: Michigan Publishing, Aug. 2013.

[Fri+12] Laurent Fribourg, David Lesens, Pierre Moro, and Romain Soulat. “Robustness Analysis for Scheduling Problems using the Inverse Method”. In: *TIME*. Ed. by Mark Reynolds, Paolo Terenziani, and Ben Moszkowski. Leicester, UK: IEEE Computer Society Press, Sept. 2012, pp. 73–80. DOI: 10.1109/TIME.2012.10.

[Lut+19] Lars Luthmann, Timo Gerecht, Andreas Stephan, Johannes Bürdek, and Malte Lochau. “Minimum/maximum delay testing of product lines with unbounded parametric real-time constraints”. In: *Journal of Systems and Software* 149 (2019), pp. 535–553. DOI: 10.1016/j.jss.2018.12.028.

[Par+16] Baptiste Parquier, Laurent Rioux, Rafik Henia, Romain Soulat, Olivier H. Roux, Didier Lime, and Étienne André. “Applying parametric model-checking techniques for reusing real-time critical systems”. In: *FTSCS*. (Nov. 14, 2016). Ed. by Cyrille Artho and Peter Csaba Ölveczky. Vol. 694. Communications in Computer and Information Science. Tokyo, Japan: Springer, Nov. 2016, pp. 129–144. DOI: 10.1007/978-3-319-53946-1_8.

[SAL15] Youcheng Sun, Étienne André, and Giuseppe Lipari. “Verification of Two Real-Time Systems Using Parametric Timed Automata”. In: *WATERS*. (July 7, 2015). Ed. by Sophie Quinton and Tullio Vardanega. Lund, Sweden, July 2015.

[WAH19] Masaki Waga, Étienne André, and Ichiro Hasuo. “Symbolic Monitoring against Specifications Parametric in Time and Data”. In: *CAV, Part I*. (July 15–18, 2019). Ed. by Işil Dillig and Serdar Tasiran. Vol. 11561. Lecture Notes in Computer Science. New York City, USA: Springer, 2019, pp. 520–539. DOI: 10.1007/978-3-030-25540-4_30.