Efficient Analysis of Parametric Hybrid Systems using HYMITATOR

Étienne André[1], Laurent Fribourg[2], Ulrich Kühne[3] and Romain Soulat[2]

[1]LIPN, CNRS UMR 7030, Université Paris 13, France
[2]ENS Cachan, CNRS, LSV, UMR8643
[3]Universität Bremen, Germany

Abstract. ⇐ to do ⇒

Keywords: Real-Time Systems, Hybrid Automata, Verification, Parameter Synthesis

⇐ **Version avec commentaires** ⇒

1 Motivation

⇐ **short introduction** ⇒

In [3], we proposed the inverse method for Timed Automata, a subclass of hybrid systems whose variables (named clocks) all have constant rates equal to 1. Different from CEGAR-based methods, this original semi-algorithm for parameter synthesis is based on a "good" parameter valuation π_0 instead of a set of "bad" states. *IM* synthesizes a constraint K_0 on the parameters such that, for all parameter valuation π satisfying K_0, the trace set, i.e., the discrete behavior, of $\mathcal{A}$ under π is the same as for $\mathcal{A}$ under π_0. This preserves in particular linear time properties, and provides the system with a criterion of robustness. By iterating the inverse method on all integer points within a bounded reference parameter domain, we get a set of constraints ("tiles") such that, for every point in each such constraint, the time-abstract behavior is the same: this gives a behavioral cartography of the system [4].

A basic implementation named IMITATOR (for *Inverse Method for Inferring Time AbstracT behaviOR*) has first been proposed, under the form of a Python script calling HYTECH [8]. The tool has then been entirely rewritten in IMITATOR II [2], under the form of a standalone OCaml program making use of the Parma Polyhedra Library (PPL) [6]. A number of case studies containing up to 60 timing parameters could be efficiently verified in the purely timed framework.

The inverse method and the behavioral cartography have then been extended to hybrid systems in [7], and implemented in a prototype "fork" of IMITATOR II. ⇐ **small description to add** ⇒ We present in this paper HYMITATOR, an extension of that prototype, performing parameter synthesis on hybrid systems.

⇐ **Laurent a dit :** ⇒ - le principe de la methode inverse pour les TA a été implémenté de façon basique (IMITATOR 1 [ICTAC09]) puis de façon sophistiquée (IMITATOR 2 [Infinity10])

- la méthode a été étendue pour les HA et implémentée de façon basique (Imitator 3 [RP11])

- il s'agit "ici" de décrire une implémentation sophistiquée (intégrant notamment des extensions du merging des TA à la [NFM 12] aux HA)

2 Architecture and Features

HYMITATOR takes as input a network of hybrid automata synchronized on shared actions. The input syntax, inspired by the input syntax of HYTECH, allows the use of analog variables (i.e. variables such as time, velocity or temperature), rational-valued discrete variables, and parameters (i.e. unknown constants). The dynamics of the analog variables is described by ordinary differential equations. The tool directly supports linear dynamics, while affine dynamics can be approximated with arbitrary precision.

The core of the program is written in the object-oriented language OCaml, and interacts with PPL. Exact arithmetics is used. A constraint is output in text format; furthermore, the set of traces computed by the analysis can be output under a graphical form (using Graphviz).

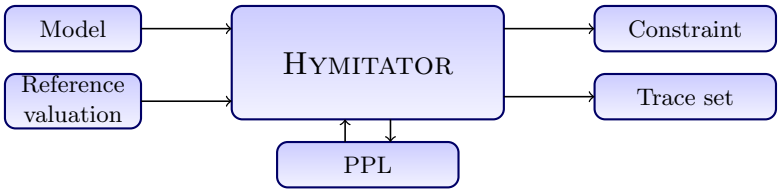

Fig. 1. Architecture of HYMITATOR

HYMITATOR implements the following algorithms for hybrid systems:

Full reachability analysis Given a model, it computes the set of symbolic reachable states.

Predicate Abstraction Safety verification can alternatively be performed using a counterexample-guided abstraction refinement loop. The abstract state space is constructed wrt. a set of linear predicates. The method has been described in [1].

Inverse method Given a model and a reference parameter valuation π_0, it computes a constraint on the parameter guaranteeing the same time-abstract behavior as under π_0

Behavioral cartography Given a model and a bounded parameter domain for each parameter valuation, it computes a set of constraints and their corresponding trace sets.

HYMITATOR uses several algorithmic optimizations, some of which have initially benn developed for IMITATOR. In particular, the efficient merging presented

in [5] has been successfully extended to the hybrid case: we merge any two states sharing the same discrete part (location and value of the discrete variables) and such that the union of their constraint on the analog variables and parameters is convex. This optimization preserves the correctness of all our algorithms; better, the constraint output by the inverse method in that case may be weaker, i.e., covers a larger set of parameter valuations.

For affine hybrid systems, further optimizations are needed. Due to the linear overapproximation by partitioning the state space, a lot of additional branching is introduced, which renders the inverse method ineffective. To solve this problem, the algorithm has been extended as described in [7]. Basically, the partitioning is performed locally, and partitions belonging to the same discrete state are merged by taking their convex hull.

⇐ un mot sur la cartographie puisque, je crois, elle a ete modifiee dans le cas des systemes hybrides ⇒

The post image computation can be costly for hybrid automata. To overcome this problem, an abstraction technique for the verification of simple safety properties (non-reachability of bad states) has been presented in [1]. Based on a set of linear predicates, reachability is performed on the abstract state space induced by these predicates. Refinement can be performed by discovering separation planes. While the original method is based on flow-pipe construction, we adapted the algorithm to the linear approximation by state space partitioning.

⇐ un mot sur l'algo CEGAR-like de Ulrich ⇒

3 Applications

⇐ Laurent a dit : ⇒

- sampled data hybrid systems (cf [7])
- test coverage for hybrid systems
- synthesis schedulability regions (cf [Cimatti-Palopoli-Ramadian08], [Astrium EADS])

⇐ experiences ? ⇒

⇐ rapport d'etudes de cas a rediger ⇒

4 Related Work

⇐ to do (citer HyTech, Phaver) ⇒

A graphical user interface for the input model is currently under construction, based on a generic platform. **⇐ utile ?! ⇒**

References

1. R. Alur, T. Dang, and F. Ivančić. Predicate abstraction for reachability analysis of hybrid systems. *ACM Trans. Embed. Comput. Syst.*, 5:152–199, February 2006.

2. É. André. IMITATOR II: A tool for solving the good parameters problem in timed automata. In *INFINITY'10*, volume 39 of *EPTCS*, pages 91–99, 2010.
3. É. André, T. Chatain, E. Encrenaz, and L. Fribourg. An inverse method for parametric timed automata. *International Journal of Foundations of Computer Science*, 20(5):819–836, 2009.
4. É. André and L. Fribourg. Behavioral cartography of timed automata. In *RP'10*, volume 6227 of *LNCS*, pages 76–90. Springer, 2010.
5. É. André, L. Fribourg, and R. Soulat. Enhancing the inverse method with state merging. In A. Goodloe and S. Person, editors, *NFM'12*, volume 7226 of *LNCS*, pages 100–105. Springer, 2012. To appear.
6. R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008.
7. L. Fribourg and U. Kühne. Parametric verification and test coverage for hybrid automata using the inverse method. In G. Delzanno and I. Potapov, editors, *RP'11*, volume 6945 of *Lecture Notes in Computer Science*, pages 191–204. Springer, 2011.
8. T. A. Henzinger, P. H. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1:460–463, 1997.

Appendix

Example of Trace Set

⇐ ajouter un trace set en mode fancy ⇒

An Example of Model

⇐ ajouter un modele en entree avec sa representation graphique (pour decrire la syntaxe) ⇒