Pithya: A Parallel Tool for Parameter Synthesis of Piecewise Multi-affine Dynamical Systems

Nikola Beneš, Luboš Brim, Martin Demko,
Samuel Pastva, and David Šafránek(✉)

Systems Biology Laboratory, Faculty of Informatics,
Masaryk University, Brno, Czech Republic
xsafran1@fi.muni.cz

Abstract. We present a novel tool for parameter synthesis of piecewise multi-affine dynamical systems from specifications expressed in a hybrid branching-time temporal logic. The tool is based on the algorithm of parallel semi-symbolic coloured model checking that extends standard model checking methods to cope with parametrised Kripke structures. The tool implements state-of-the-art techniques developed in our previous research and is primarily intended to be used for the analysis of dynamical systems with uncertain parameters that frequently arise in computational systems biology. However, it can be employed for any dynamical system where the non-linear equations can be sufficiently well approximated by piecewise multi-affine equations.

1 Introduction

Complex dynamical systems arise in many areas such as biology, biophysics, economy, or social sciences. To study them, various kinds of models are used. Such models usually employ some parameters that either represent unknown mechanics of the real-world system or serve as a way of tuning the behaviour of the system. A popular way of modelling dynamical systems is to employ the framework of differential equations with parameters. To find an analytical solution to these equations is often intractable due to the complexity of the system, the number of parameters and their interdependencies. A different approach, the one we focus on here, is to discretise the system, thereby obtaining a parametrised transition system, a kind of a computational model in the sense of [21]. Such systems are amenable to processing by formal methods. We formalise the desired properties of the system's dynamics in a suitable (temporal) logic and then, using an algorithm similar to model checking, we find parameter valuations under which the model satisfies given properties (i.e., exhibits the desired behaviour).

This work has been supported by the Czech Science Foundation grant GA15-11089S. The authors acknowledge Matej Hajnal for functional testing and language corrections.

R. Majumdar and V. Kunčak (Eds.): CAV 2017, Part I, LNCS 10426, pp. 591–598, 2017.
DOI: 10.1007/978-3-319-63387-9_29

In this paper, we present Pithya, a new parallel semi-symbolic tool for parameter synthesis. The input to our tool is a parametrised model of a continuous-time dynamical system. Currently, models represented by means of autonomous ordinary differential equations (ODEs) with sigmoidal functions are supported. This format covers most of the models commonly used in computational systems biology [22]. The model is first approximated into a piecewise multi-affine model and subsequently discretised into a parametrised direction transition system (PDTS) [7]. States of a PDTS are labelled with basic atomic propositions explicitly characterising the variable values. The transitions are indexed by parameter valuations and labelled with *directions* of change in the affected variable values. The use of directions as transition labels allows to reason about the flows in the system. To formalise the desired properties of the model we employ a hybrid extension of the UCTL logic [5] with past, called HUCTL$_{\mathrm{P}}$ [7]. Action-based parts of UCTL allow to express properties about directions; the hybrid extension together with the past/future duality of operators allows to capture interesting dynamical properties of states such as sinks, sources, cycles etc.

The parameter synthesis engine of the tool thus obtains a PDTS model and an HUCTL$_{\mathrm{P}}$ formula and its job is to compute the set of all parameter valuations under which the PDTS satisfies the formula. In order to do so, the engine employs the parallel algorithm for coloured model checking [11], more specifically, the semi-symbolic version presented in [6] (the extension to HUCTL$_{\mathrm{P}}$ was presented in [7]). The sets of parameter valuations in the PDTS are represented symbolically as first-order arithmetic formulae (while the states are represented explicitly). To deal with these formulae, the tool makes use of an SMT solver.

The algorithm starts by partitioning the PDTS into fragments [14] and distributing them among the working nodes (workstations in a cluster setting, processors in a multi-core setting). Each node then considers each state of its own fragment and each subformula of the given formula and computes the set of parameter valuations under which the subformula holds in the given state. This computation is done in a bottom-up dynamic programming fashion akin to original CTL model checking [16]. The sets of parameter valuations are represented as first-order arithmetic formulae. At specific times during the computation, the computed (symbolic) sets are exchanged among the nodes. The algorithm stops once no new information has been computed by any node since the last exchange.

2 Architecture

The Pithya tool consists of three parts: The main part consisting of several standalone executables, the graphical user interface (GUI) used for model design and result visualisation, and the command-line interface (CLI). Figure 1 depicts the architecture of Pithya and all its components. The white boxes denote input, output, and the auxiliary files while the coloured boxes denote the executables.

The *model input file* defines the input model and is written using our `.bio` format. The file declares the parameters of the model and defines the variables using ODEs with predefined sigmoidal functions. For more information about the `.bio` format see the tool manual.

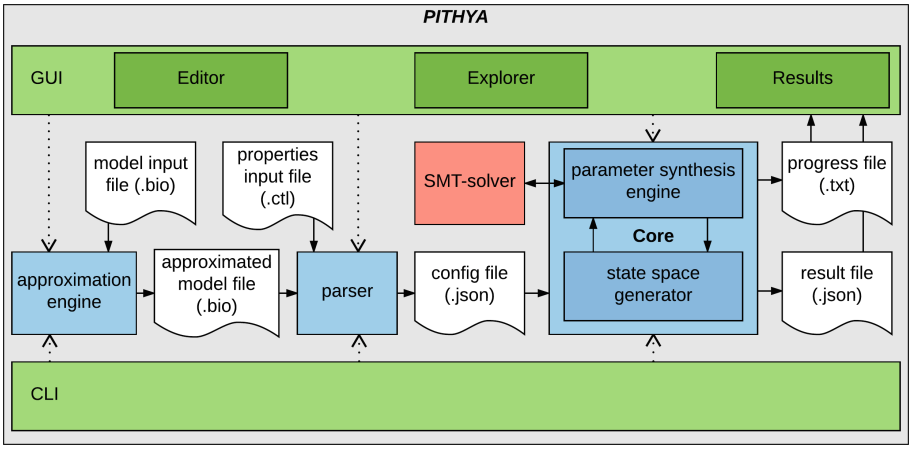

Fig. 1. The architecture of Pithya. (Colour figure online)

The *approximation engine* is a stand-alone executable that verifies the syntax of the model and performs the piecewise multi-affine approximation of the ODEs (using the approach defined in [22]). The GUI can use this executable separately to visualise the approximated model without performing the full parameter synthesis.

The *approximated model file* is an auxiliary `.bio` file produced by the approximation engine containing the piecewise multi-affine approximated version of the original model.

The *properties input file* defines properties of the model as HUCTL_P formulae in our `.ctl` format. For more information about this format see the tool manual.

The *parser* is a stand-alone executable that verifies the syntax of the provided formulae with respect to the approximated model and prepares the final configuration file used by the core executable.

The *configuration file* is an auxiliary file in the JSON format that describes both the input model together with the properties of interest in a machine-readable way that is used by the core executable.

The *core* performs the parameter synthesis based on the parallel coloured model checking approach. The core consists of a model-agnostic parameter synthesis engine together with a model-specific state space generator. The space generator employs the rectangular abstraction as defined in [4,22].

The *SMT-solver* is either an internal solver (Pithya implements a very efficient solver for models with independent parameters) or an external instance of the Z3 prover [25]. Using Z3 allows Pithya to handle models with interdependent parameters.

The *progress file* is a simple text output from the core engine through the standard error channel. It contains various useful information mostly used by the GUI to inform the user about the progress of the synthesis.

The *result file* is the main output from core engine through the standard output channel containing the results of the parameter synthesis. This includes the set of satisfying states along with the corresponding parameter valuations for each investigated property. This file is written in the JSON format that is further processed by the visualisation part of the GUI. The user may request a different, more human-readable output format. For more details about the supported output formats see the tool manual.

The *command-line interface* (CLI) encapsulates all the stand-alone executables so that the user only needs to provide the model input file and the properties file. The executables are run automatically and provide the result of the parameter synthesis in any of the supported output formats.

The *graphical user interface* (GUI) consists of three parts. The *editor* allows the user to load, edit, and save the description of the model and the properties of interest. The *explorer* is used to investigate the model behaviour and its approximated transition state space. The *results* visualiser provides an interactive visual analysis of the parameter synthesis results. For more information about the possibilities of the GUI see the tool manual.

3 Implementation

Pithya is available at http://biodivine.fi.muni.cz/pithya under the GPL license. It relies on SMT solving for the core parameter synthesis procedure, however, it does so in a way that is not entirely conventional. Instead of issuing a small number of difficult queries, Pithya iteratively builds up the knowledge about the system by issuing a high amount of simple queries while maintaining a compact symbolic representation of the intermediate results.

The effectiveness of this approach relies on several key observations:

- SMT solving is the main performance bottleneck.
- Small independent queries can be easily solved concurrently, even if the solver itself does not support parallel evaluation.
- The intermediate results can immediately influence ongoing computation, therefore merging several execution paths or cutting others entirely.
- The queries can be often simplified during the solving procedure and the size of the resulting queries does not increase substantially during the whole computation.
- The complexity of SMT solving is worse than linear. Therefore, it can be faster to iteratively issue small queries and simplify their results, even if the simplification procedure costs more than just plain solving.

Assignment Caching. Except for optimisations, another relatively cheap by-product of formula solving is often a satisfying parameter assignment. Such assignment is saved and later used to speed up solving of formulae derived from the original satisfiable formula.

Adaptive Optimiser. To achieve optimal balance between solving and simplifying, Pithya tracks the average size of the simplified formulae and adjusts the threshold for future optimisations accordingly. This ensures that the size of the formulae does not grow too much while reducing the need for costly optimisations of formulae that are already almost minimal.

4 Evaluation and Applications

The methods implemented in Pithya have been successfully employed for complex analysis in several case studies.

In [11], a well-known model of cancer-critical phase transition in mammalian cell cycle has been analysed using the prototype tool. Fully automated parameter synthesis has been used to analyse systems stability in the case of independent parameters. The achieved results are in good agreement with traditional numerical continuation analysis. In [6], the results have been extended to parameter synthesis of interdependent parameters, a very difficult task to achieve with numerical methods. When supplied with several properties the method can also be used to find the boundaries in the parametric space where the satisfiability of the properties changes. Such boundaries are called bifurcation points and the prototype has been applied [7] to complex bifurcation analysis.

In [23], the prototype has been used to explore the behaviour of various models of signalling pathways. In particular, it has been discovered under which parameter valuations the models reproduce abnormal behaviour observed in cells of organisms suffering serious illnesses such as dysplasia or cancer.

The prototype has been also applied in synthetic biology [18]. In particular, a synthetic pathway for efficient biodegradation of a toxic substance has been designed and fine-tuned with the help of parameter synthesis for the given temporal specification of desired behaviour.

Regarding the performance, the prototype has been evaluated on several different models showing that scalability of the parallel algorithm copes well with increasing number of synthesised parameters. In particular, it was possible to compute the results on multi-dimensional models (5–8 variables) for up to six parameters in tens of minutes on a common homogeneous cluster equipped with quad-core Intel Xeon 2 GHz processors [11,12].

5 Related Tools

RoVerGeNe [4] uses the same piecewise multi-affine approximation and discrete abstraction that is employed in Pithya. However, the algorithm employs different approach and heuristics to explore the parameter space. Recently, there has been developed an extension Hydentify [9] for multi-affine hybrid automata that incorporates time but is limited to reachability properties only. GNA [3] employs different approximation/abstraction techniques (piecewise affine systems) while using NuSMV as the model checker. BioPsy [24] implements parameter synthesis with respect to time-series data. It is entirely based on SMT for formulae

over reals and employs δ-decidability. The limitation is that the technique is limited only to reachability analysis. Sapo [20] implements parameter synthesis for discrete-time polynomial dynamical systems specified by difference equations and supports (linear-time) Signal Temporal Logic (STL). BioCham [26], Breach [19] and Parasim [13] employ parameter synthesis for linear-time logics. Sampling is used along with numerical methods to simulate trajectories and explore the parameters w.r.t. a given formula by computing quantitative satisfaction/robustness measures.

There are several tools for discrete models based on Boolean networks (BNs). To the best of our knowledge, the only tools that offer parameter synthesis for BNs and temporal formulae are Esther [29] and TREMPPI [28]. BMA [1] is a model checker for BNs that is based on LTL. Antelope [2] is a model checker that employs branching-time hybrid logic. ANIMO [27] uses timed automata and UPPAAL [8] as the computation engine and thus is limited to reachability.

Tools for parameter synthesis have been recently developed also in the domain of stochastic models. PROPhESY [17] supports discrete-time models and reachability properties. PRISM-PSY [15] implements parametric uniformisation for Continuous Stochastic Logic and employs GPU hardware. U-check [10] employs Bayesian statistical algorithm and smoothed model checking.

References

1. Ahmed, Z., et al.: Bringing LTL model checking to biologists. In: Bouajjani, A., Monniaux, D. (eds.) VMCAI 2017. LNCS, vol. 10145, pp. 1–13. Springer, Cham (2017). doi:10.1007/978-3-319-52234-0_1
2. Arellano, G., Argil, J., Azpeitia, E., Benítez, M., Carrillo, M., Góngora, P., Rosenblueth, D.A., Alvarez-Buylla, E.R.: "Antelope": a hybrid-logic model checker for branching-time Boolean GRN analysis. BMC Bioinf. **12**(1), 1–15 (2011)
3. Batt, G., Page, M., Cantone, I., Gössler, G., Monteiro, P., de Jong, H.: Efficient parameter search for qualitative models of regulatory networks using symbolic model checking. Bioinformatics **26**(18), 603–610 (2010)
4. Batt, G., Yordanov, B., Weiss, R., Belta, C.: Robustness analysis and tuning of synthetic gene networks. Bioinformatics **23**(18), 2415–2422 (2007)
5. ter Beek, M.H., Fantechi, A., Gnesi, S., Mazzanti, F.: A state/event-based model-checking approach for the analysis of abstract system properties. Sci. Comput. Program. **76**(2), 119–135 (2011)
6. Beneš, N., Brim, L., Demko, M., Pastva, S., Šafránek, D.: Parallel SMT-based parameter synthesis with application to piecewise multi-affine systems. In: ATVA 2016. LNCS, vol. 9936, pp. 1–17. Springer, Cham (2016)
7. Beneš, N., Brim, L., Demko, M., Pastva, S., Šafránek, D.: A model checking approach to discrete bifurcation analysis. In: Fitzgerald, J., Heitmeyer, C., Gnesi, S., Philippou, A. (eds.) FM 2016. LNCS, vol. 9995, pp. 85–101. Springer, Cham (2016). doi:10.1007/978-3-319-48989-6_6
8. Bengtsson, J., Larsen, K., Larsson, F., Pettersson, P., Yi, W.: UPPAAL — a tool suite for automatic verification of real-time systems. In: Alur, R., Henzinger, T.A., Sontag, E.D. (eds.) HS 1995. LNCS, vol. 1066, pp. 232–243. Springer, Heidelberg (1996). doi:10.1007/BFb0020949

9. Bogomolov, S., Schilling, C., Bartocci, E., Batt, G., Kong, H., Grosu, R.: Abstraction-based parameter synthesis for multiaffine systems. In: Piterman, N. (ed.) HVC 2015. LNCS, vol. 9434, pp. 19–35. Springer, Cham (2015). doi:10.1007/978-3-319-26287-1_2
10. Bortolussi, L., Milios, D., Sanguinetti, G.: U-check: model checking and parameter synthesis under uncertainty. In: Campos, J., Haverkort, B.R. (eds.) QEST 2015. LNCS, vol. 9259, pp. 89–104. Springer, Cham (2015). doi:10.1007/978-3-319-22264-6_6
11. Brim, L., Češka, M., Demko, M., Pastva, S., Šafránek, D.: Parameter synthesis by parallel coloured CTL model checking. In: Roux, O., Bourdon, J. (eds.) CMSB 2015. LNCS, vol. 9308, pp. 251–263. Springer, Cham (2015). doi:10.1007/978-3-319-23401-4_21
12. Brim, L., Demko, M., Pastva, S., Šafránek, D.: High-performance discrete bifurcation analysis for piecewise-affine dynamical systems. In: Abate, A., Šafránek, D. (eds.) HSB 2015. LNCS, vol. 9271, pp. 58–74. Springer, Cham (2015). doi:10.1007/978-3-319-26916-0_4
13. Brim, L., Vejpustek, T., Šafránek, D., Fabriková, J.: Robustness analysis for value-freezing signal temporal logic. In: HSB 2013, EPTCS, vol. 125, pp. 20–36 (2013)
14. Brim, L., Yorav, K., Žídková, J.: Assumption-based distribution of CTL model checking. STTT **7**(1), 61–73 (2005)
15. Češka, M., Pilař, P., Paoletti, N., Brim, L., Kwiatkowska, M.: PRISM-PSY: precise GPU-accelerated parameter synthesis for stochastic systems. In: Chechik, M., Raskin, J.-F. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 367–384. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49674-9_21
16. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst. **8**(2), 244–263 (1986)
17. Dehnert, C., Junges, S., Jansen, N., Corzilius, F., Volk, M., Bruintjes, H., Katoen, J.-P., Ábrahám, E.: PROPhESY: a probabilistic parameter synthesis tool. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9206, pp. 214–231. Springer, Cham (2015). doi:10.1007/978-3-319-21690-4_13
18. Demko, M., Beneš, N., Brim, L., Pastva, S., Šafránek, D.: High-performance symbolic parameter synthesis of biological models: a case study. In: Bartocci, E., Lio, P., Paoletti, N. (eds.) CMSB 2016. LNCS, vol. 9859, pp. 82–97. Springer, Cham (2016). doi:10.1007/978-3-319-45177-0_6
19. Donzé, A., Fanchon, E., Gattepaille, L.M., Maler, O., Tracqui, P.: Robustness analysis and behavior discrimination in enzymatic reaction networks. PLoS ONE **6**(9), e24246 (2011)
20. Dreossi, T.: Sapo: Reachability computation and parameter synthesis of polynomial dynamical systems. CoRR abs/1607.02200 (2016)
21. Fisher, J., Henzinger, T.: Executable cell biology. Nat. Biotechnol. **25**(11), 1239–1249 (2007)
22. Grosu, R., Batt, G., Fenton, F.H., Glimm, J., Guernic, C., Smolka, S.A., Bartocci, E.: From cardiac cells to genetic regulatory networks. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 396–411. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22110-1_31
23. Hajnal, M., Šafránek, D., Demko, M., Pastva, S., Krejčí, P., Brim, L.: Toward modelling and analysis of transient and sustained behaviour of signalling pathways. In: Cinquemani, E., Donzé, A. (eds.) HSB 2016. LNCS, vol. 9957, pp. 57–66. Springer, Cham (2016). doi:10.1007/978-3-319-47151-8_4

24. Madsen, C., Shmarov, F., Zuliani, P.: `BioPSy`: an SMT-based tool for guaranteed parameter set synthesis of biological models. In: Roux, O., Bourdon, J. (eds.) CMSB 2015. LNCS, vol. 9308, pp. 182–194. Springer, Cham (2015). doi:10.1007/978-3-319-23401-4_16
25. de Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78800-3_24
26. Rizk, A., Batt, G., Fages, F., Soliman, S.: A general computational method for robustness analysis with applications to synthetic gene networks. Bioinformatics **25**(12), i169–i178 (2009)
27. Schivo, S., Scholma, J., van der Vet, P.E., Karperien, M., Post, J.N., van de Pol, J., Langerak, R.: Modelling with animo: between fuzzy logic and differential equations. BMC Syst. Biol. **10**(1), 56 (2016)
28. Streck, A.: Toolkit for reverse engineering of molecular pathways via parameter identification. Ph.D. thesis, Free University of Berlin, Germany (2016)
29. Streck, A., Kolčák, J., Siebert, H., Šafránek, D.: Esther: introducing an online platform for parameter identification of boolean networks. In: Gupta, A., Henzinger, T.A. (eds.) CMSB 2013. LNBI, vol. 8130, pp. 257–258. Springer, Heidelberg (2013)