M2 Internship Proposal: Parameterized Synthesis via Data Word Automata, Logics and Games

Emmanuel Filiot and Pierre-Alain Reynier

Motivations: Parameterized verification and synthesis Parameterized systems arise in the practice of, for instance, distributed algorithms, telecommunication protocols, or robotics. They consist in programs which are made of an arbitrary number of processes which can communicate with each other. The distributed nature of parameterized systems makes their design hard to realize correctly, and computer-assisted methods are necessary to help their design. During an execution of a parameterized system, it is assumed that the number of processes is a parameter k which can be arbitrary but does not change over time. So, different executions may have different numbers of processes k.

To help the design of parameterized systems, various mathematical models have been introduced as well as automatic verification methods. Their goal is to automatically verify, for example, that whatever the number of processes k is, all the possible executions of those k processes satisfy a property, given for instance as a logical formula. See for instance [2] for a survey of those models and methods. Such verification techniques run over abstract models of paramaterized systems rather than their concrete instance. The models therefore should over-approximate the concrete systems so that when the verification algorithm concludes that the model satisfies a given property, so does the concrete system.

Verification techniques therefore need to take a model of a system as input, and are somehow used as high-level debugging tools. The synthesis problem is a more ambitious problem: it asks to automatically generate a parameterized system from a specification of its correct behaviours. So, in a synthesis scenario, only a set of correct semantical behaviours of the system is specified (typically in a logical formalism), and it is the goal of the synthesis algorithm to automatically construct a system whose executions are all semantically correct, if such a system exists. The automatic synthesis of programs from specifications is an active field of research which has seen substantial progress in the last decade. One successful example is the automatic synthesis of (non-parameterized) reactive systems [4]. Reactive systems are in continuous interaction with some environment which provides signals to which the system must react, while ensuring the correctness of its behaviours with respect to a set of properties typically expressed in a temporal logic. There are now many efficient tools for reactive synthesis and a yearly competition [1].

A *parameterized reactive system* can be defined as a system interacting with an environment made of an arbitrary number k of processes, and whose goal

is to ensure some property. As an example, the environment can be composed of processes which demand access to some critical resource and the goal of the system is to ensure that any process requiring access to the resource eventually gets it.

Word languages and games for reactive system synthesis A convenient abstraction for modelling reactive system executions is that of infinite words over a finite alphabet. Such words model the interaction between the environment and the system, which alternatively provide an input symbol $i \in I$ taken in a finite set I and an output symbol $o \in O$ taken in a finite set O. The result of that interaction is an infinite word $i_1 o_1 i_2 o_1 \ldots \in (IO)^\omega$. Then, correctness properties of executions are, semantically, languages $P \subseteq (IO)^\omega$: an execution $w \in (IO)^\omega$ is correct with respect to P if $w \in P$. The reactive synthesis problem (also known as the Church problem [6, 11, 4]) then asks whether given such a P, there exists a function $f : I^+ \to O$ such that for all inputs $i_1 i_2 \ldots \in I^\omega$, $i_1 f(i_1) i_2 f(i_1 i_2) \ldots \in P$. Such a function f is called a *causal* function or a *strategy* and indeed, the synthesis problem of reactive systems can be seen as a game between two antagonistic players: the environment which tries to fail the system with respect to P and the system which tries to guarantee that P holds by reacting to the inputs he received so far. A celebrated result is that of Büchi and Landweber obtained in 69: when P is defined in monadic second-order logic with one successor (or equivalently, P is ω-regular), the reactive synthesis problem is decidable. The solution of this problem is based on automata theory and games played on graphs [13].

Data word languages as a semantical abstraction of parameterized reactive systems In parameterized synthesis, the environment is composed of arbitrary many processes. The latter assumption that I and O are finite sets then appears to be a strong limitation to accurately model executions of such systems. In this internship, we rather propose to model interactions as *data words*, i.e., infinite sequences of elements taken in a infinite set (e.g. N,Q, .). For example, integers in N can be used to represent process ids. Consider the example of a system which receives, at each step of its execution, the id of a process, and eventually has to select the highest id and output it forever. Correct executions of such a system form the following data word language:

$$\mathcal{S} = \{d_1 d'_1 d_2 d'_2 \ldots \in N^\omega \mid \exists i, m \in N \cdot (\forall j \geq i \cdot d'_j = d_m) \wedge (\forall k \in N \cdot d_k \leq d_m)\}$$

A way to formulate the parameterized synthesis problem is to ask whether for all number of processes k, there exists a strategy ensuring that the executions are all in $\mathcal{S}$, whatever the environment does, i.e. whatever the input sequence amongst k different data the environment provides. The answer is positive in our example: a winning strategy would output at every step, the maximal id seen so far. This strategy satisfies the specification whatever k is, because by definition of the parameter k, at most k different id can occur as input, and so, there exists a maximal one.

There is a large literature on automata to recognize, and logics to define, data word languages. Those formalisms can thus be used to express specifications of parameterized systems, and are good candidates for a mathematical theory of synthesis of parameterized reactive systems.

Main objective The main objective of this project is to establish mathematical foundations for a theory of parameterized synthesis based on the data word abstraction. In particular, the goal is to identify classes of specifications with decidable parameterized synthesis problems, whether those specifications are defined using data word automata or data word logics, and to design synthesis algorithms able to automatically generate (models of) parameterized systems.

Work plan This research topic opens numerous research directions. As a consequence, the internship has a natural continuation as a PhD Thesis. Based on a discussion with the interested student, we will decide which research direction is investigated during the internship, the other ones being deferred to the PhD Thesis.

A first step will consist in revisiting data word languages with a parameterized perspective. More precisely, we will study different classes of register automata, and study their properties (closure properties, emptiness...) when they are considered with a new semantics, allowing only a finite (parameterized) number of distinct data values.

This preliminary study will then be used to analyse parameterized synthesis problems for various specification formalisms, either based on logics or on automata. Several classes of logics and automata can be considered to describe such specifications. For instance, various classes of register automata have been studied in [10]. Regarding the logical perspective, specifications expressed in extensions of first-order logic have recently been considered in [12]. Other logical formalisms, based on linear-time temporal logic, have been studied in [7, 8].

Regarding automata, a promising research direction consists in lifting the results obtained in [10] to this parameterized setting. We have indeed shown the decidability of different versions of the synthesis problem for specifications expressed as register automata [9]. To lift them, one will have to study whether parameterized synthesis can be reduced to classical synthesis for some classes of automata. A complementary approach consists in identifying cutoff properties. Intuitively, such a property can be thought as the existence of a finite bound K allowing to reduce all the synthesis problems for $k \geq K$ distinct data values to a single synthesis problem for exactly K distinct data values. The existence of cutoff is a classical tool for the verification parameterized systems [5].

When turning to logical aspects, the problem is again quickly undecidable, and one has to identify decidable fragments. Beyond existing restrictions of first-order logic considered in [3], one could also consider fragments of monadic-second-order logic, with suitable restrictions on input-data tests. Unlike the data-free setting, there are in general no automata-logic correspondences in presence of data. Therefore original techniques are required to address this

setting.

Thesis The internship naturally offers a continuation as a PhD thesis.

Funding Standard funding according to national rules will be offered.

Location The internship will be co-supervised by Emmanuel Filiot (ULB) and Pierre-Alain Reynier (AMU). Hence, the internship can be held either at Bruxelles, or at Marseille, and visits to the other site during the internship will be proposed.

Student profile This proposal is best fit to students interested in theoretical aspects of computer science and in particular automata theory, logics (automata-logic correspondences, model-checking, ...), as well as infinite-duration games played on graphs. The outcomes of this project will be mainly theorems and proofs and, hopefully, publications in TCS conferences.

Working language French or English.

Contact If you have any question, please contact us at **efiliot@ulb.be** or **pierre-alain.reynier@lis-lab.fr**.

References

[1] The reactive synthesis competition. www.syntcomp.com.

[2] Parosh Aziz Abdulla and Giorgio Delzanno. Parameterized verification. *Int. J. Softw. Tools Technol. Transf.*, 18(5):469–473, 2016.

[3] Béatrice Bérard, Benedikt Bollig, Mathieu Lehaut, and Nathalie Sznajder. Parameterized synthesis for fragments of first-order logic over data words. In Jean Goubault-Larrecq and Barbara König, editors, *Foundations of Software Science and Computation Structures - 23rd International Conference, FOSSACS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings*, volume 12077 of *Lecture Notes in Computer Science*, pages 97–118. Springer, 2020.

[4] Roderick Bloem, Krishnendu Chatterjee, and Barbara Jobstmann. Graph games and reactive synthesis. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 921–962. Springer, 2018.

[5] Roderick Bloem, Swen Jacobs, Ayrat Khalimov, Igor Konnov, Sasha Rubin, Helmut Veith, and Josef Widder. *Decidability of Parameterized Verification*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2015.

[6] A. Church. Logic, arithmetics, and automata. In *Proc. Int. Congress of Mathematicians, 1962*, pages 23–35. Institut Mittag-Leffler, 1963.

[7] Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3):16:1–16:30, 2009.

[8] Stéphane Demri and Karin Quaas. Concrete domains in logics: a survey. *ACM SIGLOG News*, 8(3):6–29, 2021.

[9] Léo Exibard, Emmanuel Filiot, and Pierre-Alain Reynier. Synthesis of data word transducers. *Log. Methods Comput. Sci.*, 17(1), 2021.

[10] Léo Exibard. *Automatic Synthesis of Systems with Data*. PhD thesis, Université Libre de Bruxelles, Belgium and Aix-Marseille University, France, 2021.

[11] O. Kupferman and M.Y. Vardi. Church's problem revisited. 5(2):245 – 263, 1999.

[12] Mathieu Lehaut. *Synthesis for Parameterized Systems*. PhD thesis, Sorbonne Université, 2020.

[13] Wolfgang Thomas. Church's problem and a tour through automata theory. In Arnon Avron, Nachum Dershowitz, and Alexander Rabinovich, editors, *Pillars of Computer Science, Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*, volume 4800 of *Lecture Notes in Computer Science*, pages 635–655. Springer, 2008.