History-deterministic Register Automata

Léo Exibard[1*] and Karoliina Lehtinen[2]

[1] ICE-TCS, Department of Computer Science, Reykjavik University, Iceland
leoe@ru.is

[2] CNRS, Aix-Marseille Université et Université de Toulon, LIS, Marseille, France
lehtinen@lis-lab.fr

Abstract

Recently, reactive synthesis, which asks to generate a system that guarantees correctness regardless of the behaviour of its environment, has been generalised to infinite alphabets. In this setting, specifications can be formalised as register automata. As is often the case, nondeterminism in the specification automaton leads to the synthesis problem being undecidable, yet deterministic register automata are much less expressive.

History-determinism is a restricted form of nondeterminism which combines some of the algorithmic properties of deterministic automata with some of the succinctness and expressivity of nondeterministic ones. In particular, the synthesis problem for history-deterministic automata tends to be no harder than for deterministic ones, which makes this an interesting class to consider for reactive synthesis.

In this paper, we study the expressivity and succinctness of history-deterministic register automata, as well as their whether membership to the class is decidable. We also examine whether history-determinism coincides with good-for-gameness.

1 Introduction

History-determinism While nondeterminism often makes automata models more expressive and more succinct, this power comes at a cost: many problems that are computationally easy, or at least decidable, for deterministic automata become more difficult, or even undecidable, for the corresponding nondeterministic model. This is the case for example for problems such as universality, inclusion and equivalence, which tend to be easier for deterministic models, in part thanks to their closure properties. Intermediate models, which allow some restricted form of nondeterminism, aim to combine some of the algorithmic properties of deterministic automata with some of the expressive power of nondeterminism.

History-deterministic automata [12, 5] are nondeterministic automata in which all nondeterministic choices can be made on-the-fly, without knowledge of the future of the word. This restricted nondeterminism is well-behaved with respect to composition; as a result, some computational problems are no harder for history-deterministic automata than for deterministic ones. For example, solving games with winning conditions given by a history-deterministic automaton tends to have the same complexity as solving those with deterministic winning conditions. In contrast, for nondeterministic winning conditions, solving games is either undecidable, as for pushdown automata, or involves an expensive determinisation step, as for ω-regular automata.

So far, history-determinism has mostly been studied in the ω-regular setting, where it was originally introduced by Henzinger and Piterman [12]. It then coincides with *good-for-gameness*, that is, automata for which composition preserves the winner of two-player games [3]. History-determinism and good-for-gameness are often used interchangeably even outside of the regular

*This research was supported by the project Mode(l)s of Verification and Monitorability (MoVeMnt) (no. 217987-051) of the Icelandic Research Fund.

setting; however they do not necessarily coincide [4]. For ω-regular automata, like nondeterministic automata, they have the same expressivity as deterministic ones, but can be up to exponentially more succinct [15]. In the context-free setting, they add both succinctness and expressiveness to deterministic pushdown automata, already on finite words [11], but also enjoy ExpTime-solvable universality and reactive synthesis problems [16]. History-deterministic pushdown automata however have especially poor closure properties, as they are not closed under intersection, union, complementation nor projection.

Formal Methods over Infinite Domains Recently, efforts have been made towards integrating data processing into formal methods. Various models handling data values from an infinite domain have been proposed [18, 6]. Among them, register automata [13] constitute a popular formalism for verification [7, 8, 19] and synthesis [9, 14, 10]. As an example, consider the setting of a server that has to grant requests from an a priori unbounded set of clients, where each request has to be specifically addressed to the corresponding client. Figure 1 depicts a non-deterministic register automaton that checks *violations* of this property. Register automata are also promising in runtime verification [2], to monitor properties with data dependencies.

2 Register Automata

A *data domain* consists in an infinite set $\mathbb{D}$ of *data values*, along with a finite set of predicates; typical domains are $(\mathbb{N}, =)$, $(\mathbb{Q}, \leq)$, and $(\mathbb{N}, \leq)$. Then, a finite (respectively, ω-)*data word* is a finite (resp., infinite) sequence of elements of the domain. Although this can be encoded in the domain, it is often convenient to allow the use of *labels* from a finite alphabet Σ; the automaton then reads *labelled data words*, i.e. sequences of pairs in $\Sigma \times \mathbb{D}$.

Informally, a *register automaton* consists in an (ω-)regular automaton, equipped with a finite set of *registers* that it uses to store data values and compare them. The automaton starts in some initial configuration, that consists in an initial state and a fixed valuation of the registers. Transitions areare equipped with *tests*, that consist of quantifier-free formulas over the predicates of the domain. On reading a data value, the automaton compares it with the content of its registers by checking whether it satisfies the test. It then possibly stores it in some registers, overwriting their previous content, and transitions to another state.

A non-deterministic register automaton recognises the language of all (labelled) data words that admit at least one accepting run, where acceptance is define through an ω-regular condition on states, e.g. parity. It is deterministic if it has exactly one initial state and from any configuration there exists at most one transition that can be taken.

Nondeterministic register automata (NRA) are strictly more expressive than deterministic ones (DRA) [13, Example 10], and incomparable with their dual, universal (a.k.a. co-nondeterministic) register automata (URA) [13, Example 4] (cf also Figure 1). This means that they are not closed under complement. The cost of this expressivity is algorithmic: universality [17, Theorem 5.1], reactive synthesis [10, Theorem 3.1], inclusion and equivalence are all undecidable for NRA, while they are decidable for DRA, as those are closed under intersection and complement and their emptiness is decidable [13, Theorem 1].

3 History-determinism and the Letter Game

We study history-deterministic register automata (HRA), both over finite and infinite data words. Informally, these are register automata for which accepting runs can be built on-the-fly,

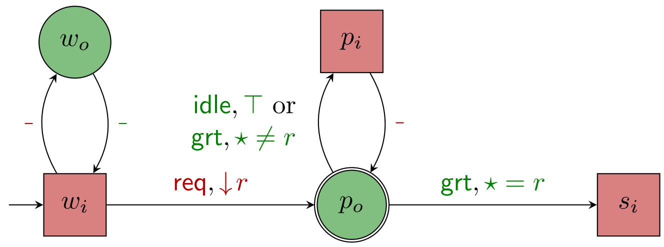

Figure 1: A non-deterministic Büchi register automaton checking that some request (label req) of some client i is never granted (label grt). $\star$ denotes the input data value, $\downarrow r$ that it is stored in register r, and _ denotes a transition that can always be taken. The initial state is w_i, and p_o is accepting. The automaton loops in w_i, until it guesses that a particular request is never granted. It stores the corresponding client ID, transitions to p_o and checks its guess: it loops infinitely often in p_o iff the request is never granted; if it is, it transitions to s_i and the run dies.

independently of the suffix of the word. This intuition can be formalised as the existence of a winning strategy for Eve in the *letter game* in which, at each turn, Adam chooses a letter from the infinite alphabet, and Eve responds with a transition of the automaton over this letter. In the limit, Eve wins if either the word w built by Adam is not in the language of the automaton, or if she built an accepting run over w. For automata over finite words, this condition has to hold at each turn. A winning strategy of Eve thus corresponds to a function $\lambda : \mathbb{D}(\Delta\mathbb{D})^* \to \Delta$ which, given a history $d_0 t_0 \dots d_n$ that corresponds to a partial run in the automaton, solves non-determinism by deciding which transition to take on reading the incoming data value d_n.

4 Results

Comparison with good-for-gameness First, for register automata over finite words and nondeterministic coBüchi register automata, the notion of history-determinism coincides with that of being good-for-games, in the sense of [12]. This is open for other acceptance conditions.

Expressivity History-deterministic RA are strictly more expressive than DRA over infinite words. Over finite words, they are determinisable by duplicating transitions and adding guards.

Decision problems and closure properties Algorithmically, they resemble DRA: inclusion, equivalence, universality, and reactive synthesis are all decidable. HRA are also closed under union and intersection, but not under complement.

Decidability of history-determinism Deciding whether an automaton is history-deterministic coincides with the good-enough synthesis problem [1] of deterministic automata of the same type [4]. This problem is decidable for register automata over finite words, thus also solving the good-enough synthesis problem of deterministic register specifications.

Open problems The case of infinite words is open, and tightly related to the determinacy and decision of games with a URA winning condition, as well as the memory structure of winning strategies: does (some form of) finite memory suffice?

References

[1] Shaull Almagor and Orna Kupferman. Good-enough synthesis. In *International Conference on Computer Aided Verification*, pages 541–563. Springer, 2020.

[2] Ezio Bartocci, Yliès Falcone, Adrian Francalanza, and Giles Reger. Introduction to runtime verification. In Ezio Bartocci and Yliès Falcone, editors, *Lectures on Runtime Verification - Introductory and Advanced Topics*, volume 10457 of *Lecture Notes in Computer Science*, pages 1–33. Springer, 2018.

[3] Udi Boker and Karoliina Lehtinen. Good for games automata: From nondeterminism to alternation. In *30th International Conference on Concurrency Theory*, 2019.

[4] Udi Boker and Karoliina Lehtinen. History-determinism and good-for-gameness in quantitative automata. In *to appear in FSTTCS'21*, 2021.

[5] Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *International Colloquium on Automata, Languages, and Programming*, pages 139–150. Springer, 2009.

[6] Loris D'Antoni. In the maze of data languages. *CoRR*, abs/1208.5980, 2012.

[7] Stéphane Demri and Deepak D'Souza. An automata-theoretic approach to constraint LTL. *Inf. Comput.*, 205(3):380–415, 2007.

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

[9] Rüdiger Ehlers, Sanjit A. Seshia, and Hadas Kress-Gazit. Synthesis with identifiers. In Kenneth L. McMillan and Xavier Rival, editors, *Verification, Model Checking, and Abstract Interpretation - 15th International Conference, VMCAI 2014, San Diego, CA, USA, January 19-21, 2014, Proceedings*, volume 8318 of *Lecture Notes in Computer Science*, pages 415–433. Springer, 2014.

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

[11] Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, and Martin Zimmermann. A Bit of Nondeterminism Makes Pushdown Automata Expressive and Succinct. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*, volume 202 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 53:1–53:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[12] Thomas A Henzinger and Nir Piterman. Solving games without determinization. In *International Workshop on Computer Science Logic*, pages 395–410. Springer, 2006.

[13] Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.

[14] Ayrat Khalimov, Benedikt Maderbacher, and Roderick Bloem. Bounded synthesis of register transducers. In Shuvendu K. Lahiri and Chao Wang, editors, *Automated Technology for Verification and Analysis - 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings*, volume 11138 of *Lecture Notes in Computer Science*, pages 494–510. Springer, 2018.

[15] Denis Kuperberg and Michał Skrzypczak. On determinisation of good-for-games automata. In *International Colloquium on Automata, Languages, and Programming*, pages 299–310. Springer, 2015.

[16] Karoliina Lehtinen and Martin Zimmermann. Good-for-games ω-pushdown automata. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '20, page 689702, New York, NY, USA, 2020. Association for Computing Machinery.

[17] Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004.

[18] Luc Segoufin. Automata and logics for words and trees over an infinite alphabet. In Zoltán Ésik, editor, *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, volume 4207 of *Lecture Notes*

in Computer Science, pages 41–57. Springer, 2006.

[19] Luc Segoufin and Szymon Torunczyk. Automata based verification over linearly ordered data domains. In Thomas Schwentick and Christoph Dürr, editors, *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany*, volume 9 of *LIPIcs*, pages 81–92. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011.