Residuality and Learning for Register Automata

Joshua Moerman

September 18, 2020

1 Introduction

Register automata (RA) are a generalisation of automata which deal with data values and data flow. Typically, the values are taken from an infinite domain, and so their languages are over an infinite alphabet. Despite that, many algorithms do generalise, since RA allow only for a restricted use of data values. Applications include reasoning about XML databases [NSV04], trace languages for analysis of programs with resource allocation [GDPT13], and behaviour of programs with data flows [HJV19].

In this research we consider the problem of inferring a register automaton from observations. This has been done before for deterministic RA [Aar14, Cas15, Fit18, MSS$^+$17], but is still open for nondeterministic RA. To see why nondeterminism is interesting, consider the well-known learning algorithms L* and NL* for respectively deterministic and nondeterministic automata. Although the representation is different, they operate on the same class of languages (i.e., *regular languages*). This is not the case for RA, where nondeterminism gives a strictly bigger class of languages than determinism. So not only does the representation changes, so does the class of languages.

Our contributions are as follows.[1]

- We consider *residual automata* for data languages. We show that their languages form a proper subclass of all languages accepted by nondeterministic RA. See Figure 1.
- We give a *machine-independent characterisation* of this class of languages. For this, we also develop some new results in nominal lattice theory.
- We show that for this class of languages, L*-style algorithms exist.

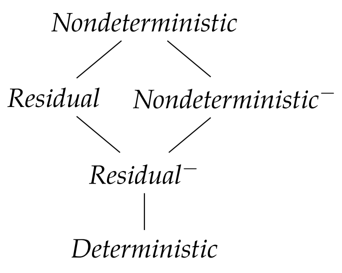


Figure 1: Different classes of data languages. With $\cdot^-$ we denote classes where automata are not allowed to *guess*.

[1]This is joint work with Matteo Sammartino, see our paper [MS20].

- The natural generalisation of NL* does not always terminate, surprisingly. Fortunately, the algorithm can be fixed to always terminate.

In particular, this settles some open problems left in [MSS+17]. We like to note that residuality was introduced in [DLT02], as a tool to better understand certain learning problems for probabilistic automata [DE08].

2 The Main Theorem

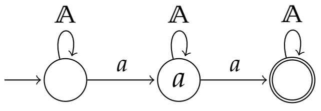
Our results are proven using *nominal automata*. The correspondence between RA and nominal automata can be found in [Boj19, BKL14]. Let $\mathbb{A}$ be a countably infinite set of atoms.

Definition 1. A *nominal automaton* is a tuple $\mathcal{A} = (Q, \Sigma, I, F, \delta)$, where Q is the state space, Σ the alphabet, $I, F \subseteq Q$ respectively initial and final states, and $\delta \subseteq Q \times \Sigma \times Q$ a transition relation. We require Q, Σ to be orbit-finite and I, F, δ to be equivariant.

Definition 2. Given a language $\mathcal{L} \subseteq \Sigma^*$ and a word $w \in \Sigma^*$, we define the *derivative* as $w^{-1}\mathcal{L} := \{u \mid wu \in \mathcal{L}\}$. The set of all derivatives of $\mathcal{L}$ is denoted by $\mathrm{Der}(\mathcal{L}) := \{w^{-1}\mathcal{L} \mid w \in \Sigma^*\}$.

An automaton $\mathcal{A}$ is *residual* if all states accept a derivative of $\mathcal{L}(\mathcal{A})$.

Example 1. The language $\mathcal{L} := \{uawav \mid u, w, v \in \mathbb{A}^*, a \in \mathbb{A}\}$ can be accepted by the automaton depicted below. The set $\mathrm{Der}(\mathcal{L})$ is not orbit-finite, hence $\mathcal{L}$ is not accepted by a deterministic automaton. However, the set is *generated* by (the orbits of) $\mathcal{L}, a^{-1}\mathcal{L}, aa^{-1}\mathcal{L}$, that is, each derivative is a union of these derivatives.



This language can be accepted by a residual automaton, precisely because of these generators. This works more generally, as our main theorem shows.

Theorem 1. *Given a language $\mathcal{L} \in \mathcal{P}(\Sigma^*)$, the following are equivalent:*

1. *$\mathcal{L}$ is accepted by a residual automaton.*
2. *There is some orbit-finite set $J \subseteq \mathrm{Der}(\mathcal{L})$ which generates $\mathrm{Der}(\mathcal{L})$.*
3. *The set $\mathcal{JI}(\mathrm{Der}(\mathcal{L}))$ is orbit-finite and generates $\mathrm{Der}(\mathcal{L})$. Here $\mathcal{JI}$ is the set of join-irreducibles.*

This characterisation is proven by constructing a *canonical* residual automaton from the join-irreducible elements. Importantly, it enables learning, since it means that a canonical automaton can be constructed from an observation table. Moreover, it shows that a *finite* observation table always exist for residual automata.

References

[Aar14] Fides Dorothea Aarts. *Tomte: bridging the gap between active learning and real-world systems*. PhD thesis, 2014.

[BKL14] Mikołaj Bojańczyk, Bartek Klin, and Slawomir Lasota. Automata theory in nominal sets. *Logical Methods in Computer Science*, 10(3), 2014.

[Boj19] Mikołaj Bojańczyk. *Slightly Infinite Sets*. Draft September 11, 2019, 2019. https://www.mimuw.edu.pl/ bojan/paper/atom-book.

[Cas15] Sofia Cassel. *Learning Component Behavior from Tests: Theory and Algorithms for Automata with Data*. PhD thesis, 2015.

[DE08] François Denis and Yann Esposito. On rational stochastic languages. *Fundam. Inform.*, 86(1-2):41–77, 2008.

[DLT02] François Denis, Aurélien Lemay, and Alain Terlutte. Residual finite state automata. *Fundam. Inform.*, 51(4):339–368, 2002.

[Fit18] Paul Fiterău-Broştean. *Active Model Learning for the Analysis of Network Protocols*. PhD thesis, 2018.

[GDPT13] Radu Grigore, Dino Distefano, Rasmus Lerchedahl Petersen, and Nikos Tzevelekos. Runtime verification based on register automata. In *TACAS*, volume 7795 of *LNCS*, pages 260–276. Springer, 2013.

[HJV19] Falk Howar, Bengt Jonsson, and Frits W. Vaandrager. Combining black-box and white-box techniques for learning register automata. In *Computing and Software Science*, volume 10000 of *LNCS*, pages 563–588. Springer, 2019.

[MS20] Joshua Moerman and Matteo Sammartino. Residual nominal automata. In *CONCUR*, volume 171 of *LIPIcs*, pages 44:1–44:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[MSS$^+$17] Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, and Michał Szynwelski. Learning nominal automata. In *POPL*, pages 613–625. ACM, 2017.

[NSV04] Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004.