RESIDUALITY AND LEARNING FOR NONDETERMINISTIC REGISTER AUTOMATA *

JOSHUA MOERMAN AND MATTEO SAMMARTINO

RTWH Aachen University, Germany
e-mail address: joshua@cs.rwth-aachen.de

Royal Holloway University of London, United Kingdom
e-mail address: matteo.sammartino@rhul.ac.uk

ABSTRACT. We are motivated by the following question: which data languages admit an active learning algorithm? This question was left open in previous work, and is particularly challenging for languages recognised by nondeterministic automata. To answer it, we develop the theory of *residual register automata*, a subclass of nondeterministic register automata. We prove that this class has canonical representatives, which can always be constructed via a finite number of observations. This property enables active learning algorithms, and makes up for the fact that residuality – a semantic property – is undecidable for register automata. Our construction for canonical residual automata is based on a machine-independent characterisation of residual languages, for which we develop new results in nominal lattice theory. Studying residuality in the context of nominal languages is a step towards a better understanding of learnability of automata with some sort of nondeterminism.

1. INTRODUCTION

Formal languages over infinite alphabets have received considerable attention recently. They include data languages for reasoning about XML databases [NSV04], trace languages for analysis of programs with resource allocation [GDPT13], and behaviour of programs with data flows [HJV19]. Typically, these languages are accepted by *register automata*, first introduced in the seminal paper [KF94].

In this paper, we investigate the properties of so-called *residual* register automata. An automaton accepting a language $\mathcal{L}$ is residual whenever the language of each state is a *derivative* of $\mathcal{L}$. In the context of regular languages over finite alphabets, residual finite state automata (RFSAs) are a subclass of nondeterministic finite automata (NFAs) introduced by Denis et al. [DLT02] as a solution to the well-known problem of NFAs *not having* unique minimal representatives. They show that every regular language admits a unique canonical RFSA, which can be much smaller than the canonical deterministic automaton.

$$\text{NONDETERMINISTIC}$$

$$\text{RESIDUAL} \quad \text{NONDETERMINISTIC}^-$$
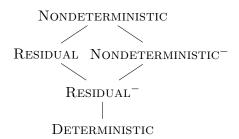
$$\text{RESIDUAL}^-$$

$$\text{DETERMINISTIC}$$

Figure 1: Relationship between classes of data languages. Edges are strict inclusions. With $\cdot^-$ we denote classes where automata are not allowed to *guess* values, i.e., to store symbols in registers without explicitly reading them.

Residual automata play a key role in the context of *exact learning*[1], in which one computes an automaton representation of an unknown language via a finite number of observations. The defining property of residual automata allows one to (eventually) observe the semantics of each state independently. In the finite-alphabet setting, residuality underlies the seminal algorithm $\mathsf{L}^\star$ for learning deterministic automata [Ang87] (deterministic automata are always residual), and enables efficient algorithms for learning nondeterministic [BHKL09] and alternating automata [AEF15, BLLR17]. Residuality has also been studied for learning probabilistic automata [DE08]. Existence of canonical residual automata is crucial for the convergence of these algorithms.

Our investigation of residuality in the context of data languages is motivated by the question: which data languages admit an exact learning algorithm? In previous work [MSS+17], we have shown that the $\mathsf{L}^\star$ algorithm generalises smoothly to data languages, meaning that deterministic register automata can be learned. However, the nondeterministic case proved to be significantly more challenging. In fact, in stark contrast with the finite-alphabet case, nondeterministic register automata are *strictly more expressive* than deterministic ones, so that residual automata are not just succinct representations of deterministic languages. As a consequence, our attempt to generalise the $\mathsf{NL}^\star$ algorithm for nondeterministic finite automata to register automata only partially succeeded: we only proved that it converges for deterministic languages, leaving the nondeterministic case open. By investigating residual data languages, we are finally able to settle this case.

We use the theory of nominal sets as our mathematical foundation to deal with register automata. The correspondence between nominal automata and register automata provided a fruitful research program in which many classical results can be generalised to register automata [BKL14, Boj19]. For example, the deterministic nominal automata admit canonical minimal models which are unique up to isomorphism, and many textbook algorithms generalise easily.

In summary, our contributions are as follows:

- Section 3: We refine classes of data languages as depicted in Figure 1, by giving separating languages for each class.
- Section 4: We develop new results of nominal lattice theory, from which we prove the main characterisation theorem (Theorem 4.10). This provides a machine-independent characterisation of the languages accepted by residual nominal automata and constructs

---

[1]Exact learning is also known as query learning or active (automata) learning [Ang87].

canonical automata which: a) are minimal in their respective class and unique up to isomorphism; b) can be constructed via a finite number of observations of the language. We also give an analogous result for non-guessing automata (Theorem 4.17).
- Section 5: We study decidability and closure properties for residual nominal automata. We prove that, like for nondeterministic nominal automata, equivalence is undecidable. On the other hand, universality is decidable for residual register automata.
- Section 6: We settle important open questions about exact learning of data languages. We show that residuality does not imply convergence of existing algorithms, and we give a (modified) NL⋆-style algorithm that works precisely for residual languages.

This research mirrors that of *residual probabilistic automata* [DE08]. There, too, one has distinct classes of which the deterministic and residual ones admit canonical automata and have an algebraic characterisation. We believe that our results contribute to a better understanding of learnability of automata with some sort of nondeterminism.

1.1. **Differences with conference version.** This paper is the extended version of [MS20], published at CONCUR'20. Since then we have added:
- full proofs for all the results;
- a new result stating that equivalence is undecidable;
- a greatly expanded Section 6, with the learning algorithm, proofs and bounds;
- more elaborate discussion in which we consider alternating and unambiguous nominal automata, as well as other symmetries.

2. Preliminaries

2.1. **Nominal Sets.** Our paper is based on the theory of nominal sets of [Pit13] in order to approach register automata.[2]

The main idea is that we consider the configurations of a register automaton as an infinite (but finitely-representable) set with symmetries. This way we can avoid defining a specific syntax for register automata and we can adopt a more abstract view to reason about their semantics directly. The abstract view still allows for algorithms and manipulations of register automata [BKL14], and we base our definitions of automata on this line of work.

Finiteness of the state space is captured by so-called *orbit-finiteness* and *finite supports* capture the fact that there are finitely many registers. We will now introduce these notions.

Let $\mathbb{A} = \{a, b, c, \ldots\}$ be a countably infinite set of *atoms* and let $\mathsf{Perm}(\mathbb{A})$ be the set of *permutations on* $\mathbb{A}$, i.e., the bijective functions $\pi \colon \mathbb{A} \to \mathbb{A}$. Permutations form a group where the unit is given by the identity function, the inverse by functional inverse, and multiplication by function composition.

A function $\cdot \colon \mathsf{Perm}(\mathbb{A}) \times X \to X$ is a *group action* if it satisfies $\pi \cdot (\pi' \cdot x) = (\pi \circ \pi') \cdot x$ and $\mathsf{id} \cdot x = x$. We often omit the $\cdot$ and write πx instead. We say that a set of atoms $A \subset \mathbb{A}$ *supports* $x \in X$ whenever $\pi x = x$ for all the permutations π that fix A pointwise (i.e., $\pi(a) = a$ for all $a \in A$). A *nominal set* is a set X together with a group action such that each $x \in X$ has a *finite* support (that is, each x is *finitely supported*). Every element of a nominal set $x \in X$ has *least support* which we denote by $\mathsf{supp}(x)$.

---

[2]One can equivalently use the theory of *sets with atoms* from [Boj19].

Given a nominal set X, the group action extends to subsets of X. Let $U \subseteq X$ be a subset, then we define the group action as $\pi \cdot U := \{\pi x \mid x \in U\}$. A subset U that is supported by the empty set is called *equivariant* and for such U we have $\pi U = U$ for all permutations π. This definition extends to relations and functions. For instance, a function $f \colon X \to Y$ between nominal sets is equivariant whenever $\pi f(x) = f(\pi x)$.

Two elements of a nominal set $x, y \in X$ can be considered equivalent if there is a permutation π such that $\pi x = y$. This defines an equivalence relation and partitions the set into classes. Such an equivalence class is called an *orbit*. Concretely, the orbit of x is given by $\mathsf{orb}(x) := \{\pi x \mid \pi \in \mathsf{Perm}(\mathbb{A})\}$. A nominal set X is *orbit-finite* whenever it is a finite union of orbits. More generally, we consider *A-orbits*, those are orbits defined over permutations fixing a finite set $A \subset \mathbb{A}$, namely $A\text{-}\mathsf{orb}(x) := \{\pi x \mid \pi \in \mathsf{Perm}(\mathbb{A}), \forall a \in A \colon \pi(a) = a\}$. We note that $(A\text{-})$orbit-finite sets are "robust", in the sense that results assuming equivariance (i.e., $A = \emptyset$) lift to an arbitrary A without much effort (see e.g. [Pit13, Chapter 5]). Orbit-finite sets are finitely-representable, hence algorithmically tractable [BBKL12].

2.2. Nominal Power Set.
Given a nominal set X, the *nominal power set* is defined as

$$\mathcal{P}_{\mathsf{fs}}(X) := \{U \subseteq X \mid U \text{ is finitely supported}\} .$$

Equivalently, one could define $\mathcal{P}_{\mathsf{fs}}(X)$ as the nominal set of all finitely supported functions of type $X \to 2$. This set is a Boolean algebra of which all the operations (union, intersection, and complement) are equivariant maps [GLP11].

We will also use the *finite power set* defined as

$$\mathcal{P}_{\mathsf{fin}}(X) := \{U \subseteq X \mid U \text{ is finite}\} .$$

This does not form a Boolean algebra, as it does not have a top element. However, it is still a join-semilattice. In Section 4 we develop more of the theory of nominal lattices.

Example 2.1. The set $\mathcal{P}_{\mathsf{fs}}(\mathbb{A})$ contains all finite subsets of $\mathbb{A}$ as well as all co-finite subsets of $\mathbb{A}$. The support of a finite set $A \subset \mathbb{A}$ is simply A itself, and the support of a co-finite set B is given by $\mathbb{A} \setminus B$. On the other hand $\mathcal{P}_{\mathsf{fin}}(\mathbb{A})$ only contains the finite subsets. We notice that $\mathcal{P}_{\mathsf{fin}}(\mathbb{A}) \subsetneq \mathcal{P}_{\mathsf{fs}}(\mathbb{A}) \subsetneq \mathcal{P}(\mathbb{A})$.

Note that neither power sets preserve orbit-finite sets. This fact often hinders generalising certain algorithms, such as the subset construction on nondeterministic automata, to the nominal setting.

2.3. Other Symmetries.
So far we have introduced nominal sets based on pure atoms $(\mathbb{A}, =)$, which only allow data values to be compared for equality. The theory of orbit-finite sets can be parametrised over other data theories. Another structure often considered is the dense linear order $(\mathbb{Q}, \leq)$, which allow data values to be compared for order. We refer to [BKL14] for other structures with applications to register automata.

In this paper we restrict to the pure atoms, because of the following useful property, used in Lemma 4.6.

Lemma 2.2. *Given a poset $(P, \leq)$ where P is a nominal set and $\leq$ is equivariant,*

(1) *if $x \leq y$ and x and y are in the same orbit, then $x = y$;*
(2) *if P is orbit-finite and non-empty, then it has a minimal element (i.e. some $m \in P$ such that $n \leq m$ implies $n = m$).*

Proof. Ad (1), let x, y in the same orbit. There is a finite permutation π such that $y = \pi x$, which gives $x \leq y = \pi x$. By equivariance of $\leq$ we get $\pi x \leq \pi y$, which gives

$$x \leq \pi x \leq \pi^2 x \leq \pi^3 x \leq \cdots$$

Since π is a finite permutation we have $\pi^k = \mathsf{id}$ for some k and so $x \leq y \leq \cdots \leq x$, proving $x = y$.

Ad (2), consider the finite set of orbits $\mathsf{Orb}(P) := \{\mathsf{orb}(x) \mid x \in P\}$ and the induced relation $\preceq$ defined by $o_1 \preceq o_2$ if there are $x \in o_1$ and $y \in o_2$ such that $x \leq y$. The relation $\preceq$ is clearly reflexive. For transitivity, consider $o_1 \preceq o_2 \preceq o_3$ with witnesses $x \leq y$ and $y' \leq z$; there is a permutation π such that $y = \pi y'$, giving $x \leq y = \pi y' \leq \pi z$, which shows $o_1 \leq o_3$. For antisymmetry, consider $o_1 \preceq o_2$ and $o_2 \preceq o_1$ with witnesses $x \leq y$ and $y' \leq x'$. Again there is a permutation π such that $y = \pi y'$ which gives $x \leq y = \pi y' \leq \pi x'$. By (1) we have $\pi x' = x$ and hence $x = y$ as required. We have shown that $(\mathsf{Orb}(P), \preceq)$ is a finite poset, therefore it has a minimal orbit o in $\mathsf{Orb}(P)$, and any element of o is minimal in P. $\qquad\square$

The above property does not hold for the ordered atoms $(\mathbb{Q}, \leq)$. To see this, consider the nominal poset $\mathbb{Q}$ itself (which is a single orbit set) with the given order. This poset has no minimal element.

2.4. **Nominal Automata.** The theory of nominal automata seamlessly extends classical automata theory by having orbit-finite nominal sets and equivariant functions in place of finite sets and functions.

Definition 2.3. A *(nondeterministic) nominal automaton* $\mathcal{A}$ consists of an orbit-finite nominal set Σ, the alphabet, an orbit-finite nominal set of states Q, and the following equivariant subsets

$$\underbrace{I \subseteq Q}_{\text{initial states}} \qquad \underbrace{F \subseteq Q}_{\text{final states}} \qquad \underbrace{\delta \subseteq Q \times \Sigma \times Q}_{\text{transitions}}.$$

The usual notions of acceptance and language apply. We denote the language accepted by $\mathcal{A}$ by $\mathcal{L}(\mathcal{A})$, and the language accepted by a state q by $\mathcal{L}(q)$. Note that the language $\mathcal{L}(\mathcal{A}) \in \mathcal{P}_{\mathsf{fs}}(\Sigma^*)$ is equivariant, and that $\mathcal{L}(q) \in \mathcal{P}_{\mathsf{fs}}(\Sigma^*)$ need not be equivariant, but it is supported by $\mathsf{supp}(q)$.

Remark 2.4. In most examples we take the alphabet to be $\Sigma = \mathbb{A}$, but it can be any orbit-finite nominal set. For instance, $\Sigma = \mathsf{Act} \times \mathbb{A}$, where Act is a finite set of actions, represents actions $\mathsf{act}(x)$ with one parameter $x \in \mathbb{A}$ (actions with arity n can be represented via n-fold products of $\mathbb{A}$).

We recall the notion of *derivative language* [DLT02].[3]

Definition 2.5. Given a language $\mathcal{L}$ and a word $u \in \Sigma^*$, we define the *derivative of $\mathcal{L}$ with respect to u* as

$$u^{-1}\mathcal{L} := \{w \mid uw \in \mathcal{L}\}$$

and the set of all derivatives as

$$\mathsf{Der}(\mathcal{L}) := \left\{u^{-1}\mathcal{L} \mid u \in \Sigma^*\right\}.$$

---

[3]This is sometimes called a *residual language* or *left quotient*. We do not use the term residual language here, because residual language will mean a language accepted by a residual automaton.

These definitions seamlessly extend to the nominal setting. Note that $w^{-1}\mathcal{L}$ is finitely supported whenever $\mathcal{L}$ is.

Of special interest are the deterministic, residual, and non-guessing nominal automata, which we introduce next.

Definition 2.6. A nominal automaton $\mathcal{A}$ is:
- *Deterministic* if $I = \{q_0\}$, and for each $q \in Q$ and $a \in \Sigma$ there is a unique q' such that $(q, a, q') \in \delta$. In this case, the relation is in fact functional $\delta \colon Q \times \Sigma \to Q$.
- *Residual* if each state $q \in Q$ accepts a derivative of $\mathcal{L}(\mathcal{A})$, formally: $\mathcal{L}(q) = w^{-1}\mathcal{L}(\mathcal{A})$ for some word $w \in \Sigma^*$. The words w such that $\mathcal{L}(q) = w^{-1}\mathcal{L}(\mathcal{A})$ are called *characterising words* for the state q.
- *Non-guessing* if $\mathsf{supp}(q_0) = \emptyset$, for each $q_0 \in I$, and $\mathsf{supp}(q') \subseteq \mathsf{supp}(q) \cup \mathsf{supp}(a)$, for each $(q, a, q') \in \delta$.

Observe that the transition function of a deterministic automaton preserves supports (i.e., if C supports (q, a) then C also supports $\delta(q, a)$). Consequently, all deterministic automata are non-guessing. For the sake of succinctness, in the following we drop the qualifier "nominal" when referring to these classes of nominal automata.

For many examples, it is useful to define the notion of an anchor. Given a state q, a word w is an *anchor* if $\delta(I, w) = \{q\}$, that is, the word w leads to q and no other state. Every anchor for q is also a characterising word for q (but not vice versa). A state with an anchor is called *anchored*, and we call an automaton *anchored* if all states have anchors.

Finally, we recall the Myhill-Nerode theorem for nominal automata.

Theorem 2.7 ([BKL14, Theorem 5.2])**.** *Let $\mathcal{L}$ be a language. Then $\mathcal{L}$ is accepted by a deterministic automaton if and only if $\mathsf{Der}(\mathcal{L})$ is orbit-finite.*

3. Separating languages

Deterministic, nondeterministic and residual automata have the same expressive power when dealing with finite alphabets. The situation is more nuanced in the nominal setting. We now give one language for each class in Figure 1. For simplicity, we mostly use the one-orbit nominal set of atoms $\mathbb{A}$ as alphabet. These languages separate the different classes, meaning that they belong to the respective class, but not to the classes below or beside it.

For each example language $\mathcal{L}$, we depict: a nominal automaton recognising $\mathcal{L}$ (on the left); the set of derivatives $\mathsf{Der}(\mathcal{L})$ (on the right). We make explicit the poset structure of $\mathsf{Der}(\mathcal{L})$: grey rectangles represent orbits of derivatives, and lines stand for set inclusions (we grey out irrelevant ones). This poset may not be orbit-finite, in which case we depict a small, indicative part. Observing the poset structure of $\mathsf{Der}(\mathcal{L})$ explicitly is important for later, where we show that the existence of residual automata depends on it. We write $aa^{-1}\mathcal{L}$ to mean $(aa)^{-1}\mathcal{L}$. Variables $a, b, \ldots$ are always atoms and $u, w, \ldots$ are always words.

Deterministic: First symbol equals last symbol. Consider the language
$$\mathcal{L}_{\mathrm{d}} := \{awa \mid a \in \mathbb{A}, w \in \mathbb{A}^*\}.$$

This is accepted by the following deterministic nominal automaton (Figure 2). The automaton is actually infinite-state, but we represent it symbolically using a register-like notation, where we annotate each state with the current value of a register. Note that the

derivatives $a^{-1}\mathcal{L}_{\mathrm{d}}, b^{-1}\mathcal{L}_{\mathrm{d}}, \dots$ are in the same orbit. In total $\mathsf{Der}(\mathcal{L}_{\mathrm{d}})$ has three orbits, which correspond to the three orbits of states in the deterministic automaton. The derivative $awa^{-1}\mathcal{L}_{\mathrm{d}}$, for example, equals $aa^{-1}\mathcal{L}_{\mathrm{d}}$.

Figure 2: A deterministic automaton accepting $\mathcal{L}_{\mathrm{d}}$, and the poset $\mathsf{Der}(\mathcal{L}_{\mathrm{d}})$.

Non-guessing residual: Some atom occurs twice. The language is

$$\mathcal{L}_{\mathrm{ng,r}} \coloneqq \{uavaw \mid u,v,w \in \mathbb{A}^*, a \in \mathbb{A}\}.$$

The poset $\mathsf{Der}(\mathcal{L}_{\mathrm{ng,r}})$ is not orbit-finite, so by the nominal Myhill-Nerode theorem there is no deterministic automaton accepting $\mathcal{L}_{\mathrm{ng,r}}$. However, derivatives of the form $ab^{-1}\mathcal{L}_{\mathrm{ng,r}}$ can be written as a union $ab^{-1}\mathcal{L}_{\mathrm{ng,r}} = a^{-1}\mathcal{L}_{\mathrm{ng,r}} \cup b^{-1}\mathcal{L}_{\mathrm{ng,r}}$. In fact, we only need an orbit-finite set of derivatives to recover $\mathsf{Der}(\mathcal{L}_{\mathrm{ng,r}})$. These orbits are highlighted in the diagram on the right (Figure 3). Selecting the "right" derivatives is the key idea behind constructing residual automata in Theorem 4.10.
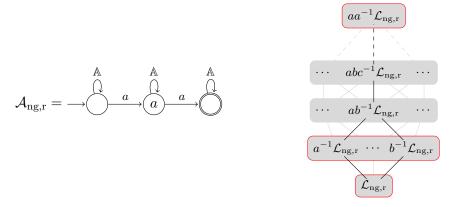
Figure 3: A nondeterministic automaton accepting $\mathcal{L}_{\mathrm{ng,r}}$, and the poset $\mathsf{Der}(\mathcal{L}_{\mathrm{ng,r}})$. The depicted automaton is not residual, but a residual automaton exists by Theorem 4.10.

Nondeterministic: Last letter is unique. We consider the language

$$\mathcal{L}_{\mathrm{n}} \coloneqq \{wa \mid a \text{ not in } w\} \cup \{\epsilon\}.$$

Derivatives $a^{-1}\mathcal{L}_{\mathrm{n}}$ are again unions of smaller languages: $a^{-1}\mathcal{L}_{\mathrm{n}} = \bigcup_{b \neq a} ab^{-1}\mathcal{L}_{\mathrm{n}}$. However, the poset $\mathsf{Der}(\mathcal{L})$ has an infinite descending chain of languages (with an increasing support), namely $a^{-1}\mathcal{L} \supset ab^{-1}\mathcal{L} \supset abc^{-1}\mathcal{L} \supset \dots$ Figure 4 shows this descending chain, where we have omitted languages like $aa^{-1}\mathcal{L}_{\mathrm{n}}$, as they only differ from $a^{-1}\mathcal{L}_{\mathrm{n}}$ on the empty word. The existence of a such a chain implies that $\mathcal{L}_{\mathrm{n}}$ cannot be accepted by a residual automaton. This is a consequence of Theorem 4.10, as we shall see later.
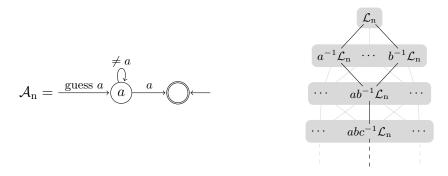
Figure 4: A nondeterministic automaton accepting $\mathcal{L}_{\mathrm{n}}$, and the poset $\mathsf{Der}(\mathcal{L}_{\mathrm{n}})$.

Residual: Last letter is unique but anchored. We reconsider the previous automaton $\mathcal{A}_{\mathrm{n}}$ and add a transition in order to make the automaton residual. First, we extend the alphabet to $\Sigma = \mathbb{A} \cup \{\mathsf{Anc}(a) \mid a \in \mathbb{A}\}$, where Anc is nothing more than a label. Then, we add the transitions $(a, \mathsf{Anc}(a), a)$ for each $a \in \mathbb{A}$, see Figure 5. The language accepted by the new automaton is

$$\mathcal{L}_{\mathrm{r}} = \{wa \mid a \in \mathbb{A}, w \in (\mathbb{A} \cup \{\mathsf{Anc}(a)\})^*\}$$

Here, we have forced the automaton to be residual, by adding an anchor to the first state. Nevertheless, guessing is still necessary. In the poset, we note that all elements in the descending chain can now be obtained as unions of $\mathsf{Anc}(a)^{-1}\mathcal{L}_{\mathrm{r}}$. For instance, $a^{-1}\mathcal{L}_{\mathrm{r}} = \bigcup_{b \neq a} \mathsf{Anc}(b)^{-1}\mathcal{L}_{\mathrm{r}}$. Note that $\mathsf{Anc}(a)\mathsf{Anc}(b)^{-1}\mathcal{L}_{\mathrm{r}} = \emptyset$ and $\mathsf{Anc}(a)a^{-1}\mathcal{L}_{\mathrm{r}} = \{\epsilon\}$.
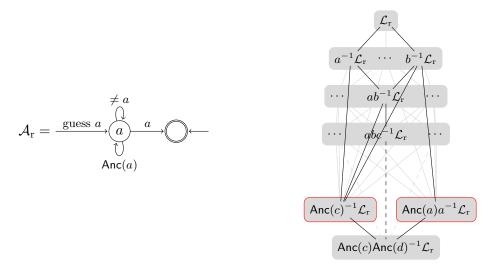

Figure 5: A residual automaton accepting $\mathcal{L}_{\mathrm{r}}$, and the poset $\mathsf{Der}(\mathcal{L}_{\mathrm{r}})$.

Non-guessing nondeterministic: Repeated atom with different successor. Consider the language

$$\mathcal{L}_{\mathrm{ng}} \coloneqq \{uabvac \mid u, v \in \mathbb{A}^*, a, b, c \in \mathbb{A}, b \neq a\}.$$

Here, we allow $a = b$ or $a = c$ in this definition. This is a language which can be accepted by a non-guessing automaton (Figure 6). However, there is no residual automaton for this language. The poset structure of $\mathsf{Der}(\mathcal{L}_{\mathrm{ng}})$ is very complicated. We will return to this example after Theorem 4.10.
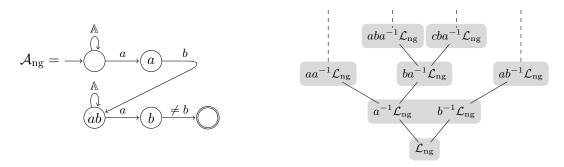

Figure 6: A deterministic automaton accepting $\mathcal{L}_{\mathrm{ng}}$, and the poset $\mathsf{Der}(\mathcal{L}_{\mathrm{ng}})$.

4. Canonical Residual Nominal Automata

In this section we will give a characterisation of *canonical* residual automata. We will first introduce notions of nominal lattice theory, then we will state our main result (Theorem 4.10). We conclude the section by providing similar results for non-guessing automata.

4.1. **Theory of Nominal Join-Semilattices.** We abstract away from words and languages and consider the set $\mathcal{P}_{\mathsf{fs}}(Z)$ for an arbitrary nominal set Z. This is a Boolean algebra of which the operations $\wedge, \vee, \neg$ are all equivariant maps [GLP11]. Moreover, the finitely supported union

$$\bigvee \colon \mathcal{P}_{\mathsf{fs}}(\mathcal{P}_{\mathsf{fs}}(Z)) \to \mathcal{P}_{\mathsf{fs}}(Z)$$

is also equivariant. We note that this is more general than a binary union, but it is not a complete join-semilattice. Hereafter, we shall denote set inclusion by $\leq$ ($<$ when strict).

Definition 4.1. Given a nominal set Z and $X \subseteq \mathcal{P}_{\mathsf{fs}}(Z)$ equivariant[4], we define the set generated by X as

$$\langle X \rangle := \left\{ \bigvee \mathfrak{x} \mid \mathfrak{x} \subseteq X \text{ finitely supported} \right\} \subseteq \mathcal{P}_{\mathsf{fs}}(Z).$$

Remark 4.2. The set $\langle X \rangle$ is closed under the operation $\bigvee$, and moreover is the smallest equivariant set closed under $\bigvee$ containing X. In other words, $\langle - \rangle$ defines a closure operator. We will often say "X generates Y", by which we mean $Y \subseteq \langle X \rangle$.

Definition 4.3. Let $X \subseteq \mathcal{P}_{\mathsf{fs}}(Z)$ equivariant and $x \in X$, we say that x is *join-irreducible in X* if it is non-empty and

$$x = \bigvee \mathfrak{x} \quad \implies \quad x \in \mathfrak{x},$$

for every finitely supported $\mathfrak{x} \subseteq X$. The subset of all join-irreducible elements is denoted by

$$\mathsf{JI}(X) := \{ x \in X \mid x \text{ join-irreducible in } X \}.$$

---

[4]A similar definition could be given for finitely supported X. In fact, all results in this section generalise to finitely supported. But we use equivariance for convenience.

This is again an equivariant set. For convenience, we may use the following equivalent definition of join-irreducible: x is non-empty and $\forall x_0 \in \mathfrak{x}.x_0 < x \implies \bigvee \mathfrak{x} < x$.

Remark 4.4. In lattice and order theory, join-irreducible elements are usually defined only for a lattice (see, e.g., [DP02]). However, we define them for arbitrary subsets of a lattice. (Note that a subset of a lattice is merely a poset.) This generalisation will be needed later, when we consider the poset $\mathsf{Der}(\mathcal{L})$ which is not a lattice, but is contained in the lattice $\mathcal{P}_{\mathsf{fs}}(\Sigma^*)$.

Remark 4.5. The notion of join-irreducible, as we have defined here, corresponds to the notion of *prime* in some papers on learning nondeterministic automata [BHKL09, DLT02, MSS$^+$17]. Unfortunately, the word *prime* has a slightly different meaning in lattice theory. We stick to the terminology of lattice theory.

If a set Y is well-behaved, then its join-irreducible elements will actually generate the set Y. This is normally proven with a descending chain condition. We first restrict our attention to orbit-finite sets. The following Lemma extends [DP02, Lemma 2.45] to the nominal setting.

Lemma 4.6. *Let $X \subseteq \mathcal{P}_{\mathsf{fs}}(Z)$ be an orbit-finite and equivariant set.*
(1) *Let $a \in X, b \in \mathcal{P}_{\mathsf{fs}}(Z)$ and $a \not\leq b$. Then there is $x \in \mathsf{JI}(X)$ such that $x \leq a$ and $x \not\leq b$.*
(2) *Let $a \in X$, then $a = \bigvee \{x \in X \mid x \text{ join-irreducible in } X \text{ and } x \leq a\}$.*

Proof. *Ad 1.* Consider the set $S = \{x \in X \mid x \leq a, x \not\leq b\}$. This is a finitely supported and $\mathsf{supp}(S)$-orbit-finite set, hence it has some minimal element $m \in S$ by Lemma 2.2 (here we are using its generalisation to $\mathsf{supp}(S)$-orbit finite sets). We shall prove that m is join-irreducible in X. Let $\mathfrak{x} \subseteq X$ finitely supported and assume that $x_0 < m$ for each $x_0 \in \mathfrak{x}$. Note that $x_0 < m \leq a$ and so that $x_0 \notin S$ (otherwise m was not minimal). Hence $x_0 \leq b$ (by definition of S). So $\bigvee \mathfrak{x} \leq b$ and so $\bigvee \mathfrak{x} \notin S$, which concludes that $\bigvee \mathfrak{x} \neq m$, and so $\bigvee \mathfrak{x} < m$ as required.

Ad 2. Consider the set $T = \{x \in \mathsf{JI}(X) \mid x \leq a\}$. This set is finitely supported, so we may define the element $b = \bigvee T \in \mathcal{P}_{\mathsf{fs}}(Z)$. It is clear that $b \leq a$, we shall prove equality by contradiction. Suppose $a \not\leq b$, then by *(1.)*, there is a join-irreducible x such that $x \leq a$ and $y \not\leq b$. By the first property of x we have $x \in T$, so that $x \not\leq b = \bigvee T$ is a contradiction. We conclude that $a = b$, i.e. $a = \bigvee T$ as required. $\qquad\square$

Corollary 4.7. *Let $X \subseteq \mathcal{P}_{\mathsf{fs}}(Z)$ be an orbit-finite equivariant subset. The join-irreducibles of X generate X, i.e., $X \subseteq \langle \mathsf{JI}(X) \rangle$.*

So far, we have defined join-irreducible elements relative to some fixed set. We will now show that these elements remain join-irreducible when considering them in a bigger set, as long as the bigger set is generated by the smaller one. This will later allow us to talk about *the* join-irreducible elements.

Lemma 4.8. *Let $Y \subseteq X \subseteq \mathcal{P}_{\mathsf{fs}}(Z)$ equivariant and suppose that $X \subseteq \langle \mathsf{JI}(Y) \rangle$. Then $\mathsf{JI}(Y) = \mathsf{JI}(X)$.*

Proof. ($\supseteq$) Let $x \in X$ be join-irreducible in X. Suppose that $x = \bigvee \mathfrak{y}$ for some finitely supported $\mathfrak{y} \subseteq Y$. Note that also $\mathfrak{y} \subseteq X$ Then $x = y_0$ for some $y_0 \in \mathfrak{y}$, and so x is join-irreducible in Y.

($\subseteq$) Let $y \in Y$ be join-irreducible in Y. Suppose that $y = \bigvee \mathfrak{x}$ for some finitely supported $\mathfrak{x} \subseteq X$. Note that every element $x \in \mathfrak{x}$ is a union of elements in $\mathsf{JI}(Y)$ (by the assumption

$X \subseteq \langle \mathsf{JI}(Y_0) \rangle \rangle$). Take $\mathfrak{y}_x = \{y \in \mathsf{JI}(Y) \mid y \leq x\}$, then we have $x = \bigvee \mathfrak{y}_x$ and

$$y = \bigvee \mathfrak{x} = \bigvee \left\{ \bigvee \mathfrak{y}_x \mid x \in \mathfrak{x} \right\} = \bigvee \{y_0 \mid y_0 \in \mathfrak{y}_x, x \in \mathfrak{x}\}.$$

The last set is a finitely supported subset of Y, and so there is a y_0 in it such that $y = y_0$. Moreover, this y_0 is below some $x_0 \in \mathfrak{x}$, which gives $y_0 \leq x_0 \leq y$. We conclude that $y = x_0$ for some $x_0 \in \mathfrak{x}$. $\square$

In other words, the join-irreducibles of X are the smallest set generating X.

Corollary 4.9. *If an orbit-finite set Y generates X, then $\mathsf{JI}(X) \subseteq Y$.*

4.2. **Characterising Residual Languages.** We are now ready to state and prove the main theorem of this paper. We fix the alphabet Σ. Recall that the nominal Myhill-Nerode theorem tells us that a language is accepted by a deterministic automaton if and only if $\mathsf{Der}(\mathcal{L})$ is orbit-finite. Here, we give a similar characterisation for languages accepted by residual automata. Moreover, the following result gives a canonical construction.

Theorem 4.10. *Given a language $\mathcal{L} \in \mathcal{P}_{\mathsf{fs}}(\Sigma^*)$, the following are equivalent:*

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

Proof. We prove three implications:

($1 \Rightarrow 2$). Take the set of languages accepted by the states: $J := \{\mathcal{L}(q) \mid q \in \mathcal{A}\}$. This is clearly orbit-finite, since Q is. Moreover, each derivative is generated as follows:

$$w^{-1}\mathcal{L} = \bigvee \{\mathcal{L}(q) \mid q \in \delta(I, w)\}.$$

($2 \Rightarrow 3$). We can apply Lemma 4.8 with $Y = J$ and $X = \mathsf{Der}(\mathcal{L})$. Now it follows that $\mathsf{JI}(\mathsf{Der}(\mathcal{L}))$ is orbit-finite (since it is a subset of J) and generates $\mathsf{Der}(\mathcal{L})$.

($3 \Rightarrow 1$). We can construct the following residual automaton, whose language is exactly $\mathcal{L}$:

$$Q := \mathsf{JI}(\mathsf{Der}(\mathcal{L}))$$
$$I := \left\{w^{-1}\mathcal{L} \in Q \mid w^{-1}\mathcal{L} \leq \mathcal{L}\right\}$$
$$F := \left\{w^{-1}\mathcal{L} \in Q \mid \epsilon \in w^{-1}\mathcal{L}\right\}$$
$$\delta(w^{-1}\mathcal{L}, a) := \left\{v^{-1}\mathcal{L} \in Q \mid v^{-1}\mathcal{L} \leq wa^{-1}\mathcal{L}\right\}$$

First, note that $\mathcal{A} := (\Sigma, Q, I, F, \delta)$ is a well-defined nominal automaton. In fact, all the components are orbit-finite, and equivariance of $\leq$ implies equivariance of δ. $\mathcal{L}(q) = w^{-1}\mathcal{L}$.

$$\epsilon \in \mathcal{L}(w^{-1}\mathcal{L}) \iff w^{-1}\mathcal{L} \in F \iff \epsilon \in w^{-1}\mathcal{L}$$

$$au \in \mathcal{L}(w^{-1}\mathcal{L}) \iff u \in \mathcal{L}\left(\delta(w^{-1}\mathcal{L}, a)\right)$$

$$\iff u \in \mathcal{L}\left(\{v^{-1}\mathcal{L} \in Q \mid v^{-1}\mathcal{L} \leq wa^{-1}\mathcal{L}\}\right)$$

$$\overset{(i)}{\iff} u \in \bigvee \{v^{-1}\mathcal{L} \in Q \mid v^{-1}\mathcal{L} \leq wa^{-1}\mathcal{L}\}$$

$$\iff \exists v^{-1}\mathcal{L} \in Q \text{ with } v^{-1}\mathcal{L} \leq wa^{-1}\mathcal{L} \text{ and } u \in v^{-1}\mathcal{L}$$

$$\overset{(ii)}{\iff} u \in wa^{-1}\mathcal{L} \iff au \in w^{-1}\mathcal{L}$$

At step (i) we have used the induction hypothesis (u is a shorter word than au) and the fact that $\mathcal{L}(-)$ preserves unions. At step (ii, right-to-left) we have used that $v^{-1}\mathcal{L}$ is join-irreducible. The other steps are unfolding definitions.

Finally, note that $\mathcal{L} = \bigvee \{w^{-1}\mathcal{L} \mid w^{-1}\mathcal{L} \leq \mathcal{L}\}$, since the join-irreducible languages generate all languages. In particular, the initial states (together) accept $\mathcal{L}$. $\qquad \square$

Corollary 4.11. *The construction above defines a* canonical *residual automaton with the following uniqueness property: it has the minimal number of orbits of states and the maximal number of orbits of transitions.*

Proof. State minimality follows from Corollary 4.9, where we note that the states of any residual automata accepting $\mathcal{L}$ form a generating subset of $\mathsf{Der}(\mathcal{L})$. Maximality of transitions follows from the fact that no transitions can be added without changing the language. $\qquad \square$

For finite alphabets, the classes of languages accepted by DFAs and NFAs are the same (by determinising an NFA). This means that $\mathsf{Der}(\mathcal{L})$ is always finite if $\mathcal{L}$ is accepted by an NFA, and we can always construct the canonical RFSA. Here, this is not the case, that is why we need to stipulate (in Theorem 4.10) that the set $\mathsf{JI}(\mathsf{Der}(\mathcal{L}))$ is orbit-finite *and* actually generates $\mathsf{Der}(\mathcal{L})$. Either condition may fail, as we will see in Example 4.13.

Example 4.12. In this example we show that residual automata can also be used to compress deterministic automata. The language $\mathcal{L} := \{abb\ldots b \mid a \neq b\}$ can be accepted by a deterministic automaton of 4 orbits, and this is minimal. (A zero amount of bs is also accepted in $\mathcal{L}$.) The minimal residual automaton, however, has only 2 orbits, given by the join-irreducible languages:

$$\epsilon^{-1}\mathcal{L} = \{abb\ldots b \mid a \neq b\}$$

$$ab^{-1}\mathcal{L} = \{bb\ldots b\} \qquad (a, b \in \mathbb{A} \text{ distinct})$$

The trick in defining the automaton is that the a-transition from $\epsilon^{-1}\mathcal{L}$ to $ab^{-1}\mathcal{L}$ *guesses* the value b. In the next section (Section 4.3), we will define the canonical *non-guessing* residual automaton, which has 3 orbits.

Example 4.13. We return to the examples $\mathcal{L}_{\mathrm{n}}$ and $\mathcal{L}_{\mathrm{ng}}$ from Section 3. We claim that neither language can be accepted by a residual automaton.

For $\mathcal{L}_{\mathrm{n}}$ we note that there is an infinite descending chain of derivatives

$$\mathcal{L}_{\mathrm{n}} > a^{-1}\mathcal{L}_{\mathrm{n}} > ab^{-1}\mathcal{L}_{\mathrm{n}} > abc^{-1}\mathcal{L}_{\mathrm{n}} > \cdots$$

Each of these languages can be written as a union of smaller derivatives. For instance, $a^{-1}\mathcal{L}_n = \bigcup_{b \neq a} ab^{-1}\mathcal{L}_n$. This means that $\mathsf{JI}(\mathsf{Der}(\mathcal{L}_n)) = \emptyset$, hence it does not generate $\mathsf{Der}(\mathcal{L}_n)$ and by Theorem 4.10 there is no residual automaton.

In the case of $\mathcal{L}_{ng}$, we have an infinite ascending chain

$$\mathcal{L}_{ng} < a^{-1}\mathcal{L}_{ng} < ba^{-1}\mathcal{L}_{ng} < cba^{-1}\mathcal{L}_{ng} < \cdots$$

This in itself is not a problem: the language $\mathcal{L}_{ng,r}$ also has an infinite ascending chain. However, for $\mathcal{L}_{ng}$, none of the languages in this chain are a union of smaller derivatives, which we shall now prove formally.

Claim 4.14. *All the languages in this chain are join-irreducible.*

Proof. Consider the word $w = a_k \ldots a_1 a_0$ with $k \geq 1$ and all a_i distinct atoms. We will prove that $w^{-1}\mathcal{L}_{ng}$ is join-irreducible in $\mathsf{Der}(\mathcal{L}_{ng})$, by considering all $u^{-1}\mathcal{L}_{ng} \subseteq w^{-1}\mathcal{L}_{ng}$.

Observe that if u is a suffix of w, then $u^{-1}\mathcal{L}_{ng} \subseteq w^{-1}\mathcal{L}_{ng}$. This is easily seen from the given automaton, since it may skip any prefix. We now show that u being a suffix of w is also a necessary condition.

First, suppose that u contains an atom a different from all a_i. If it is the last symbol of u, then $aaa_0 \in u^{-1}\mathcal{L}_{ng}$, but $aaa_0 \notin w^{-1}\mathcal{L}_{ng}$. If a is succeeded by b (not necessarily distinct), then either aa or aa_0 is in $u^{-1}\mathcal{L}_{ng}$. But neither aa nor aa_0 is in $w^{-1}\mathcal{L}_{ng}$. This shows that $u^{-1}\mathcal{L}_{ng} \subseteq w^{-1}\mathcal{L}_{ng}$ implies $u \in \{a_0, \ldots, a_k\}^*$.

Now, if $u = a_i$ with $i > 0$, then $a_i a_i a_{i-1} \in u^{-1}\mathcal{L}_{ng}$, but that word is not in $w^{-1}\mathcal{L}_{ng}$. Hence if $u^{-1}\mathcal{L}_{ng} \subseteq w^{-1}\mathcal{L}_{ng}$ and $|u| \leq 1$, then u is a suffix of w.

Last, we prove the same for $|u| \geq 2$. We first consider which bigrams may occur in u. Suppose that u contains a bigram $a_i a_j$ with $i > 0$ and $j \neq i - 1$. Then $a_i a_{i-1}$ is in $u^{-1}\mathcal{L}_{ng}$, but not in $w^{-1}\mathcal{L}_{ng}$. Suppose that u contains $a_0 a_i$ $(i > 0)$ or $a_0 a_0$, then $u^{-1}\mathcal{L}_{ng}$ contains either $a_0 a_0$ or $a_0 a_1$ respectively. Neither of these words are in $w^{-1}\mathcal{L}_{ng}$. This shows that $u^{-1}\mathcal{L}_{ng} \subseteq w^{-1}\mathcal{L}_{ng}$ implies that u may only contain the bigrams $a_i a_{i-1}$. In particular, these bigrams compose in a unique way. So u is a (contiguous) subword of w, whenever $u^{-1}\mathcal{L}_{ng} \subseteq w^{-1}\mathcal{L}_{ng}$.

Continuing, suppose that u ends in the bigram $a_{i+1} a_i$ with $i > 0$. Then we have $a_i a_i a_{i-1}$ in $u^{-1}\mathcal{L}_{ng}$, but not in $w^{-1}\mathcal{L}_{ng}$. This shows that u has to end in $a_1 a_0$. That is, for $u^{-1}\mathcal{L}_{ng} \subseteq w^{-1}\mathcal{L}_{ng}$ with $|u| \geq 2$, we necessarily have that u is a suffix of w.

So far, we have shown that

$$\{u \mid u^{-1}\mathcal{L}_{ng} \subseteq w^{-1}\mathcal{L}_{ng}\} = \{u \mid u \text{ is a suffix of } w\}.$$

To see that $w^{-1}\mathcal{L}_{ng}$ is indeed join-irreducible, we consider the join $X = \bigvee\{u^{-1}\mathcal{L}_{ng} \mid u \text{ is a strict suffix of } w\}$. Note that $a_k a_k \notin X$, but $a_k a_k \in w^{-1}\mathcal{L}_{ng}$. We conclude that $w^{-1}\mathcal{L}_{ng} \neq \bigvee\{u^{-1}\mathcal{L}_{ng} \mid u^{-1}\mathcal{L}_{ng} \subsetneq w^{-1}\mathcal{L}_{ng}\}$ as required. $\square$

This result implies that the set $\mathsf{JI}(\mathsf{Der}(\mathcal{L}_{ng}))$ is *not orbit-finite*. By Theorem 4.10, we can conclude that there is no residual automaton accepting $\mathcal{L}_{ng}$.

Remark 4.15. For arbitrary (nondeterministic) languages there is also a characterisation in the style of Theorem 4.10. Namely, $\mathcal{L}$ is accepted by an automaton iff there is an orbit-finite set $Y \subseteq \mathcal{P}_{fs}(\Sigma^*)$ which generates the derivatives. However, note that the set Y need not be a subset of the set of derivatives. In these cases, we do not have a canonical construction for the automaton. Different choices for Y define different automata and there is no way to pick Y naturally.

4.3. **Automata without guessing.** We reconsider the above results for non-guessing automata. Nondeterminism in nominal automata allows naturally for guessing, meaning that the automaton may store symbols in registers without explicitly reading them. However, the original definition of register automata in [KF94] does not allow for guessing, and non-guessing automata remain actively researched [MQ19]. Register automata with guessing were introduced in [KZ10], because it was realised that non-guessing automata are not closed under reversal.

To adapt to non-guessing automata, we redefine join-irreducible elements. As we would like to remove states which can be written as a "non-guessing" union of other states, we only consider joins of sets of elements where all elements are supported by the same support.

Definition 4.16. Let $X \subseteq \mathcal{P}_{\mathsf{fs}}(Z)$ be equivariant and $x \in X$, we say that x is *join-irreducible$^-$ in X* if $x = \bigvee \mathfrak{x} \implies x \in \mathfrak{x}$, for every finitely supported $\mathfrak{x} \subseteq X$ such that $\mathsf{supp}(x_0) \subseteq \mathsf{supp}(x)$, for each $x_0 \in \mathfrak{x}$. The set of all join-irreducible$^-$ elements is denoted by

$$\mathsf{JI}^-(X) \coloneqq \left\{ x \in X \mid x \text{ join-irreducible}^- \text{ in } X \right\}.$$

The only change required is an additional condition on the elements and supports in $\mathfrak{x}$. In particular, the sets $\mathfrak{x}$ are *uniformly supported* sets. Unions of such sets are called *uniformly supported unions*.

All the lemmas from the previous section are proven similarly. We state the main result for non-guessing automata.

Theorem 4.17. *Given a language $\mathcal{L} \in \mathcal{P}_{\mathsf{fs}}(\Sigma^*)$, the following are equivalent:*

(1) $\mathcal{L}$ *is accepted by a non-guessing residual automaton.*
(2) *There is some orbit-finite set $J \subseteq \mathsf{Der}(\mathcal{L})$ which generates $\mathsf{Der}(\mathcal{L})$ by uniformly supported unions.*
(3) *The set $\mathsf{JI}^-(\mathsf{Der}(\mathcal{L}))$ is orbit-finite and generates $\mathsf{Der}(\mathcal{L})$ by uniformly supported unions.*

Proof. The proof is similar to that of Theorem 4.10. However, we need a slightly different definition of the canonical automaton. It is defined as follows.

$$Q \coloneqq \mathsf{JI}^-(\mathsf{Der}(\mathcal{L}))$$
$$I \coloneqq \left\{ w^{-1}\mathcal{L} \in Q \mid w^{-1}\mathcal{L} \leq \mathcal{L}, \mathsf{supp}(w^{-1}\mathcal{L}) \subseteq \mathsf{supp}(\mathcal{L}) \right\}$$
$$F \coloneqq \left\{ w^{-1}\mathcal{L} \in Q \mid \epsilon \in w^{-1}\mathcal{L} \right\}$$
$$\delta(w^{-1}\mathcal{L}, a) \coloneqq \left\{ v^{-1}\mathcal{L} \in Q \mid v^{-1}\mathcal{L} \leq wa^{-1}\mathcal{L}, \mathsf{supp}(v^{-1}\mathcal{L}) \subseteq \mathsf{supp}(wa^{-1}\mathcal{L}) \right\}$$

Note that, in particular, the initial states have empty support since $\mathcal{L}$ is equivariant. This means that the automaton cannot guess any values at the start. Similarly, the transition relation does not allow for guessing. $\square$

To better understand the structure of the canonical non-guessing residual automaton, we recall the following fact and its consequence on non-guessing automata.

Lemma 4.18. *Let X be an orbit-finite nominal set and $A \subset \mathbb{A}$ be a finite set of atoms. The set $\{x \in X \mid A \text{ supports } x\}$ is finite.*

Corollary 4.19. *The transition relation δ of non-guessing automata can be equivalently be described as a function $\delta \colon Q \times \Sigma \to \mathcal{P}_{\mathsf{fin}}(Q)$, where $\mathcal{P}_{\mathsf{fin}}(Q)$ is the set of finite subsets of Q.*

In particular, this shows that the canonical non-guessing residual automaton has finite nondeterminism. It also shows that it is sufficient to consider *finite unions* in Theorem 4.17, instead of uniformly supported unions.

5. DECIDABILITY AND CLOSURE RESULTS

In this section we investigate decidability and closure properties of residual automata. First, a positive result: universality is decidable for residual automata. This is in contrast to the nondeterministic case, where universality is undecidable, even for non-guessing automata [Boj19].

In the constructions below, we use *computation with atoms*. This is a computation paradigm which allow algorithmic manipulation of infinite – but orbit-finite – nominal sets. For instance, it allows looping over such a set in finite time. Important here is that this paradigm is equivalent to regular computability (see [BT18]) and implementations exist to compute with atoms [KS16, KT17].

Proposition 5.1. *Universality for residual nominal automata is decidable. Formally: given a residual automaton $\mathcal{A}$, it is decidable whether $\mathcal{L}(\mathcal{A}) = \Sigma^*$.*

Proof. We will sketch an algorithm that, given a residual automaton $\mathcal{A}$, answers whether $\mathcal{L}(\mathcal{A}) = \Sigma^*$. The algorithm decides *negatively* in the following cases:

- $I = \emptyset$. In this case the language accepted by $\mathcal{A}$ is empty.
- Suppose there is a $q \in Q$ with $q \notin F$. By residuality we have $\mathcal{L}(q) = w^{-1}\mathcal{L}(\mathcal{A})$ for some w. Note that q is not accepting, so that $\epsilon \notin w^{-1}\mathcal{L}(\mathcal{A})$. Put differently: $w \notin \mathcal{L}(\mathcal{A})$. (We note that w is not used by the algorithm. It is only needed for the correctness.)
- Suppose there is a $q \in Q$ and $a \in \Sigma$ such that $\delta(q, a) = \emptyset$. Again $\mathcal{L}(q) = w^{-1}\mathcal{L}(\mathcal{A})$ for some w. Note that a is not in $\mathcal{L}(q)$. This means that wa is not in the language.

When none of these three cases hold, the algorithm decides *positively*. We shall prove that this is indeed the correct decision. If none of the above conditions hold, then $I \neq \emptyset$, $Q = F$, and for all $q \in Q, a \in \Sigma$ we have $\delta(q, a) \neq \emptyset$. Here we can prove that the language of each state is $\mathcal{L}(q) = \Sigma^*$. Given that there is an initial state, the automaton accepts Σ^*.

Note that the operations on sets performed in the above cases all terminate, because all involve orbit-finite sets. $\qquad\square$

Next we consider equivalence of residual automata and checking whether an automaton is residual. Both will turn out to be undecidable and we use following construction in order to prove this.

Construction 5.2. Let $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ be a nondeterministic automaton. Let

$$\Sigma' := \Sigma \cup \{\underline{q} \mid q \in Q\} \cup \{q \mid q \in Q\}$$

be an extended alphabet, where we assume the new symbols $\underline{q}$ and q to be disjoint from Σ. We now construct two residual automata from $\mathcal{A}$, where those symbols are used as anchors:

$$\mathcal{A}^{\mathsf{anc}} = (\Sigma', Q^{\mathsf{anc}}, I^{\mathsf{anc}}, F^{\mathsf{anc}}, \delta^{\mathsf{anc}}), \text{ where} \qquad \mathcal{A}^\top = (\Sigma', Q^\top, I^\top, F^\top, \delta^\top), \text{ where}$$

$$Q^{\mathsf{anc}} = Q \cup \{\underline{q} \mid q \in Q\} \qquad\qquad Q^\top = Q^{\mathsf{anc}} \cup \{\top\}$$

$$I^{\mathsf{anc}} = I \cup \{\underline{q} \mid q \in Q\} \qquad\qquad I^\top = \{\top\} \cup \{\underline{q} \mid q \in Q\}$$

$$F^{\mathsf{anc}} = F \qquad\qquad F^\top = F \cup \{\top\}$$

$$\delta^{\mathsf{anc}} = \delta \cup \{(\underline{q}, \underline{q}, \underline{q}) \mid q \in Q\} \qquad\qquad \delta^\top = \delta^{\mathsf{anc}} \cup \{(\top, a, \top) \mid a \in \Sigma\}$$

$$\cup \{(\underline{q}, q, q) \mid q \in Q\}$$

Note that these constructions are effective, as they involve computations over orbit-finite sets. We observe the following facts about $\mathcal{A}^{\mathsf{anc}}$ and $\mathcal{A}^\top$:

(1) The states $\underline{q}$ and q (in both automata) are anchored by the words $\underline{q}$ and q respectively. Moreover, any symbol from the original alphabet $a \in \Sigma$ is a characterising word for the state $\top$ in $\mathcal{A}^\top$. So we conclude that both automata are residual.

(2) For $q \in Q$ we note that the languages $\mathcal{L}(q^{\mathsf{anc}})$ and $\mathcal{L}(q^\top)$ are the same (where $q^{\mathsf{anc}} \in Q^{\mathsf{anc}}$ and $q^\top \in Q^\top$ denote the "same" state). Similarly we have $\mathcal{L}(\underline{q}^{\mathsf{anc}}) = \mathcal{L}(\underline{q}^\top)$.

(3) If we restrict the alphabet to the original alphabet, we get

$$\mathcal{L}(\mathcal{A}^{\mathsf{anc}}) \cap \Sigma^* = \mathcal{L}(\mathcal{A}) \qquad \text{and} \qquad \mathcal{L}(\mathcal{A}^\top) \cap \Sigma^* = \Sigma^*.$$

Proposition 5.3. *Equivalence of residual automata is undecidable.*

Proof. We show undecidability by reducing the universality problem for nondeterministic automata to the equivalence problem. (Note that a reduction from universality of residual automata will not work as that is decidable.) We use the above construction and prove that $\mathcal{A}$ is universal if and only if $\mathcal{A}^{\mathsf{anc}}$ and $\mathcal{A}^\top$ are equivalent.

Suppose $\mathcal{L}(\mathcal{A}) = \Sigma^*$, then we get $\mathcal{L}(\mathcal{A}^{\mathsf{anc}}) \cap \Sigma^* = \mathcal{L}(\mathcal{A}) = \Sigma^* = \mathcal{L}(\mathcal{A}^\top) \cap \Sigma^*$ by (3). When considering all words on Σ'^*, we note that the anchors will lead to single states q which accept the same languages by (2). So $\mathcal{L}(\mathcal{A}^{\mathsf{anc}}) = \mathcal{L}(\mathcal{A}^\top)$ as required.

Conversely, suppose that $\mathcal{L}(\mathcal{A}^{\mathsf{anc}}) = \mathcal{L}(\mathcal{A}^\top)$. Then by (3) we can conclude that

$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}^{\mathsf{anc}}) \cap \Sigma^* = \mathcal{L}(\mathcal{A}^\top) \cap \Sigma^* = \Sigma^*.$$

We conclude that we can decide universality of nondeterministic automata via equivalence of residual automata. So equivalence of residual automata is undecidable. $\qquad\square$

Last, determining whether an automaton is actually residual is undecidable. In other words, residuality cannot be characterised as a syntactic property. This adds value to learning techniques, as they are able to provide automata that are residual by construction.

Proposition 5.4. *The problem of determining whether a given nondeterministic nominal automaton is residual is undecidable.*

Proof. The construction is inspired by [DLT02, Proposition 8.4].[5] We show undecidability by reducing the universality problem for nominal automata to the residuality problem.

---

[5]They prove that checking residuality for NFAs is PSPACE-complete via a reduction from universality. Instead of using NFAs, they use a union of n DFAs. This would not work in the nominal setting.
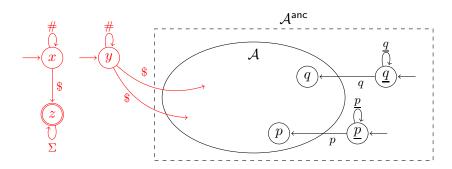
Figure 7: Sketch of the automaton $\mathcal{A}'$ constructed in the proof of Proposition 5.4.

Let $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ be a nominal (nondeterministic) automaton on the alphabet Σ. We apply Construction 5.2 and extend the alphabet Σ' further by

$$\Sigma'' := \Sigma' \cup \{\$, \#\},$$

where we assume $\{\$, \#\}$ to be disjoint from Σ. We define $\mathcal{A}' = (\Sigma', Q', I', F', \delta')$ by

$$Q' = Q^{\mathsf{anc}} \cup \{x, y, z\}$$
$$I' = I^{\mathsf{anc}} \cup \{x, y\}$$
$$F' = F^{\mathsf{anc}} \cup \{z\}$$
$$\delta' = \delta^{\mathsf{anc}} \cup \{(x, \$, z), (x, \#, x), (y, \#, y)\} \cup \{(z, a, z) \mid a \in \Sigma\} \cup \{(y, \$, q) \mid q \in I\}$$

See Figure 7 for a sketch of the automaton $\mathcal{A}'$. This is built by adding the red part to $\mathcal{A}^{\mathsf{anc}}$. The key players are states x and y with their languages $\mathcal{L}(y) \subseteq \mathcal{L}(x)$. Note that their languages are equal if and only if $\mathcal{A}$ is universal.

Before we assume anything about $\mathcal{A}$, let us analyse $\mathcal{A}'$. In particular, let us consider whether the residuality property holds for each state. From (1) we know that this holds for $\mathcal{A}^{\mathsf{anc}}$. For the states x and z we have $\mathcal{L}(z) = \Sigma^* = \$^{-1}\mathcal{L}(\mathcal{A}')$ and $\mathcal{L}(x) = \#^{-1}\mathcal{L}(\mathcal{A}')$ (see Figure 7). The only remaining state for which we do not yet know whether the residuality property holds is state y.

If $\mathcal{L}(\mathcal{A}) = \Sigma^*$ (i.e. the original automaton is universal), then we note that $\mathcal{L}(y) = \mathcal{L}(x)$. In this case, $\mathcal{L}(y) = \#^{-1}\mathcal{L}(\mathcal{A}')$. So, in this case, $\mathcal{A}'$ is residual.

Suppose that $\mathcal{A}'$ is residual. Then $\mathcal{L}(y) = w^{-1}\mathcal{L}'$ for some word w. Provided that $\mathcal{L}(\mathcal{A})$ is not empty, there is some $u \in \mathcal{L}(\mathcal{A})$. So we know that $\$u \in \mathcal{L}(y)$. This means that word w cannot start with $a \in \Sigma$, q, $\underline{q}$ for $q \in Q$, or $\$$ as their derivatives do not contain $\$u$. The only possibility is that $w = \#^k$ for some $k > 0$. This implies $\mathcal{L}(y) = \mathcal{L}(x)$, meaning that the language of $\mathcal{A}$ is universal.

This proves that $\mathcal{A}$ is universal iff $\mathcal{A}'$ is residual. $\qquad\square$

These results also hold for the subclass of non-guessing automata, as the constructions do not introduce any guessing and universality for non-guessing nondeterministic nominal automata is undecidable.

Closure properties. We will now show that several closure properties fail for residual languages. Interestingly, this parallels the situation for probabilistic languages: residual ones are not even closed under convex sums. We emphasise that residual automata were devised for learning purposes, where closure properties play no significant role. In fact, one

typically exploits closure properties of the wider class of nondeterministic models, e.g., for automata-based verification. The following results show that in our setting this is indeed unavoidable.

Consider the alphabet $\Sigma = \mathbb{A} \cup \{\mathsf{Anc}(a) \mid a \in \mathbb{A}\}$ and the residual language $\mathcal{L}_{\mathrm{r}}$ from Section 3. We consider a second language $\mathcal{L}_2 = \mathbb{A}^*$ which can be accepted by a deterministic (hence residual) automaton. We have the following non-closure results:

Union: The language $\mathcal{L} = \mathcal{L}_{\mathrm{r}} \cup \mathcal{L}_2$ cannot be accepted by a residual automaton. In fact, although derivatives of the form $\mathsf{Anc}(a)^{-1}\mathcal{L}$ are still join-irreducible (see Section 3, residual case), they have no summand $\mathbb{A}^*$, which means that they cannot generate $a^{-1}\mathcal{L} = \mathbb{A}^* \cup \bigcup_{b \neq a} \mathsf{Anc}(b)^{-1}\mathcal{L}$. By Theorem 4.10(3) it follows that $\mathcal{L}$ is not residual.

Intersection: The language $\mathcal{L} = \mathcal{L}_{\mathrm{r}} \cap \mathcal{L}_2 = \mathcal{L}_{\mathrm{n}}$ cannot be accepted by a residual automaton, as we have seen in Section 3.

Concatenation: The language $\mathcal{L} = \mathcal{L}_2 \cdot \mathcal{L}_{\mathrm{r}}$ cannot be accepted by a residual automaton, for similar reasons as the union.

Reversal: The language $\{aw \mid a \text{ not in } w\}$ is residual (even deterministic), but its reverse language is $\mathcal{L}_{\mathrm{n}}$ and cannot be accepted by a residual automaton.

Complement: Consider the language $\mathcal{L}_{\mathrm{ng,r}}$ of words where some atom occurs twice. Its complement $\overline{\mathcal{L}_{\mathrm{ng,r}}}$ is the language of all fresh atoms, which cannot even be recognised by a nondeterministic nominal automaton [BKL14].

Closure under Kleene star is yet to be settled.

5.1. **Length of characterising words.** We end this section by giving a result about the length of characterising words. Note that in the finite case, the characterising words of an n-state residual automaton have length at most 2^n, since one can determinise automata. In our case, this no longer holds, and we show that the length of characterising word is not bounded in the number of states only.

Proposition 5.5. *There is a family of residual register automata $\mathcal{A}_k$ ($k \geq 1$) with two states and k registers of which the characterising words have length k.*

Proof. We define a variation on the automaton $\mathcal{A}_{\mathrm{r}}$ from Section 3 using the alphabet $\Sigma = \mathbb{A} \cup \{\mathsf{Anc}(a) \mid a \in \mathbb{A}\}$. The automaton $\mathcal{A}_k$ is defined by the following sets, where $\binom{\mathbb{A}}{k}$ denotes the set of k-element subsets of $\mathbb{A}$.

$$Q = \binom{\mathbb{A}}{k} \cup \{\top\} \qquad I = \binom{\mathbb{A}}{k} \qquad F = \{\top\}$$

$$\delta = \left\{ (S, \mathsf{Anc}(a), S) \,\Big|\, S \in \binom{\mathbb{A}}{k}, a \in S \right\}$$

$$\cup \left\{ (S, a, S) \,\Big|\, S \in \binom{\mathbb{A}}{k}, a \notin S \right\}$$

$$\cup \left\{ (S, a, \top) \,\Big|\, S \in \binom{\mathbb{A}}{k}, a \in S \right\}$$

A state $S = \{a_1, \ldots, a_k\} \in \binom{\mathbb{A}}{k}$ is anchored by the word $w = \mathsf{Anc}(a_1) \ldots \mathsf{Anc}(a_k)$, which is of length k. This is also the shortest characterising word for that state. Note that Q only has two orbits, meaning that a register automaton equivalent to this nominal automaton only requires two states. $\qquad \square$

6. Exact learning

In our previous paper on learning nominal automata [MSS$^+$17], we provided a learning algorithm to learn residual automata, that converges for deterministic languages. However, we observed by experimentation that the algorithm was also able to learn certain nondeterministic languages. At that point we did not know which class of languages could be accepted by residual nominal automata, and so it was left open whether the algorithm converges for all residual languages. In this section we will answer this question negatively, but also provide a modified algorithm which does always converge.

6.1. **Angluin-style learning.** We briefly review the classical automata learning algorithms $L^\star$ by Angluin [Ang87] for deterministic automata, and $NL^\star$ by Bollig et al. [BHKL09] for residual automata.

Both algorithms can be seen as a game between two players: *the learner* and *the teacher*. The learner aims to construct the minimal automaton for an unknown language $\mathcal{L}$ over a finite alphabet Σ. In order to do this, it may ask the teacher, who knows about the language, two types of queries:

Membership query: Is a given word w in the target language, i.e., $w \in \mathcal{L}$?

Equivalence query: Does a given *hypothesis* automaton $\mathcal{H}$ recognise the target language, that is, is $\mathcal{L} = \mathcal{L}(\mathcal{H})$?

If the teacher replies *yes* to an equivalence query, then the algorithm terminates, as the hypothesis $\mathcal{H}$ is correct. Otherwise, the teacher must supply a *counterexample*, that is a word in the symmetric difference of $\mathcal{L}$ and $\mathcal{L}(\mathcal{H})$. Availability of equivalence queries may seem like a strong assumption and in fact it is often weakened by allowing only random sampling (see [KV94] or [Vaa17] for details).

Observations about the language made by the learner via queries are stored in an *observation table T*. This is a table where rows and columns range over two finite sets of words $S, E \subseteq \Sigma^\star$ respectively, and $T(u, v) = 1$ if and only if $uv \in \mathcal{L}$. Intuitively, each row of T approximates a derivative of $\mathcal{L}$, in fact we have $T(u) \subseteq u^{-1}\mathcal{L}$. However, the information contained in T may be incomplete: some derivatives $w^{-1}\mathcal{L}$ are not reached yet because no membership queries for w have been posed, and some pairs of rows $T(u), T(v)$ may seem equal to the learner, because no word has been seen yet which distinguishes them. The learning algorithm will add new words to S when new derivatives are discovered, and to E when words distinguishing two previously identical derivatives are discovered.

The table T is *closed* whenever one-letter extensions of derivatives are already in the table, i.e., T has a row for $ua^{-1}\mathcal{L}$, for all $u \in S, a \in \Sigma$. If the table is closed,[6] $L^\star$ is able to construct an automaton from T, where states are distinct rows (i.e., derivatives). The construction follows the classical one for the canonical automaton of a language from its derivatives [Ner58]. The $NL^\star$ algorithm uses a modified notion of closedness, where one is allowed to take unions (i.e., a one-letter extension can be written as unions of rows in T), and hence is able to learn a RFSA accepting the target language.

When the table is not closed, then a derivative is missing, and a corresponding row needs to be added. Once an automaton is constructed, it is submitted in an equivalence query. If a counterexample is returned, then again the table is extended, after which the

---

[6]$L^\star$ also needs the table to be *consistent*. We do not need that in our discussion here.

process is repeated iteratively. The $\mathsf{L}^\star$ and $\mathsf{NL}^\star$ algorithms adopt different counterexample-handling strategies: the former adds a new row, the latter a new column. Both result in a new derivative being detected.

6.2. **The nominal case.** In [MSS$^+$17] we have given nominal versions of $\mathsf{L}^\star$ and $\mathsf{NL}^\star$, called $\nu\mathsf{L}^\star$ and $\nu\mathsf{NL}^\star$ respectively. They seamlessly extend the original algorithms by operating on orbit-finite sets. This allows us to learn automata over infinite alphabets, but using only finitely many queries. The algorithm $\nu\mathsf{L}^\star$ always terminates for deterministic languages, because the language only has orbit-finitely many distinct derivatives (Theorem 2.7), and hence only need orbit-finitely many distinct rows in the observation table. However, it will *never* terminate for languages not accepted by deterministic automata (such as residual or nondeterministic languages).
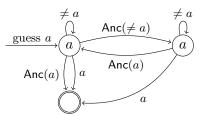
Theorem 6.1 ([Moe19]). *$\nu\mathsf{L}^\star$ converges if and only if $\mathsf{Der}(\mathcal{L})$ is orbit-finite, in which case it outputs the canonical deterministic automaton accepting $\mathcal{L}$. Moreover, at most $\mathcal{O}(nk)$ equivalence queries are needed, where n is the number of orbits of the minimal deterministic automaton, and k is the maximum support size of its states.*

The nondeterministic case is more interesting. Using Theorem 4.10, we can finally establish which nondeterministic languages can be characterised via orbit-finitely many observations.

Corollary 6.2 (of Theorem 4.10). *Let $\mathcal{L}$ be a nondeterministic nominal language. If $\mathcal{L}$ is a residual language, then there exists an observation table with orbit-finitely many rows and columns from which we can construct the canonical residual automaton.*

This explains why in [MSS$^+$17] $\nu\mathsf{NL}^\star$ was able to learn some residual nondeterministic automata: an orbit-finite observation table exists, which allows $\nu\mathsf{NL}^\star$ to construct the canonical residual automaton. Unfortunately, the $\nu\mathsf{NL}^\star$ algorithm does not guarantee that it always finds this orbit-finite observation table. We only have that guarantee for deterministic languages. The following example shows that $\nu\mathsf{NL}^\star$ may indeed diverge when trying to close the table.

Example 6.3. Suppose $\nu\mathsf{NL}^\star$ tries to learn the residual language $\mathcal{L}$ accepted by the automaton below over the alphabet $\Sigma = \mathbb{A} \cup \{\mathsf{Anc}(a) \mid a \in \mathbb{A}\}$. This is a slight modification of the residual language of Section 3.

The algorithm starts by considering the row for the empty word ϵ, and its one-letter extensions $\epsilon \cdot a = a$ and $\epsilon \cdot \mathsf{Anc}(a) = \mathsf{Anc}(a)$. These rows correspond to the derivatives $\epsilon^{-1}\mathcal{L} = \mathcal{L}$, $a^{-1}\mathcal{L}$ and $\mathsf{Anc}(a)^{-1}\mathcal{L}$. Column labels are initialised to the empty word ϵ. At this point $a^{-1}\mathcal{L}$ and $\mathsf{Anc}(a)^{-1}\mathcal{L}$ appear identical, as the only column ϵ does not distinguish them. However, they appear different from $\epsilon^{-1}\mathcal{L}$, so the algorithm will add the row for either a or $\mathsf{Anc}(a)$ in order to close the table. Suppose the algorithm decides to add a. Then it will consider one-letter extensions ab, abc, $abcd$, etc... Since these correspond to different derivatives –

each strictly smaller than the previous one – the algorithm will get stuck in an attempt to close the table. At no point it will try to close the table with the word $\mathsf{Anc}(a)$, since it stays equivalent to a. So in this case $\nu\mathsf{NL}^\star$ will not terminate. However, if the algorithm instead adds $\mathsf{Anc}(a)$ to the row labels, it will then also add $\mathsf{Anc}(a)\mathsf{Anc}(b)$, which is a characterising word for the initial state. In that case, $\nu\mathsf{NL}^\star$ will terminate.

6.3. **Modified $\nu\mathsf{NL}^\star$.** We modify the $\nu\mathsf{NL}^\star$ algorithm from [MSS$^+$17] to ensure that it always terminates. We do this by changing how the table will be closed. The algorithm is shown in Algorithm 1 with the changes made to $\nu\mathsf{NL}^\star$ in red. In short, the change is as follows. When the algorithm adds a word w to the set of rows, then it also adds all other words of length $|w|$. Since all words of bounded length are added, the algorithm will eventually find all characterising words of the canonical residual automaton, and it will therefore be able to reconstruct this automaton.

Algorithm 1 Modified nominal $\mathsf{NL}^\star$ algorithm for Theorem 6.6.

MODIFIED $\nu\mathsf{NL}^\star$ LEARNER

1   Initialise T with $S, E = \{\epsilon\}$
2   **repeat**
3   **while** T is not join-closed or not join-consistent
4   **if** T is not join-closed
5   find $s \in S, a \in A$ such that $\mathsf{row}(sa) \in \mathsf{JI}(\mathsf{Rows}(T)) \setminus \mathsf{Rows}^\uparrow(T)$
6   l = length of the word sa
7            $S = S \cup \Sigma^{\leq l}$
8   **if** T is not join-consistent
9   find $s_1, s_2 \in S$, $a \in A$, and $e \in E$ such that
               $\mathsf{row}(s_1) \leq \mathsf{row}(s_2)$ but $e \in \mathsf{row}(s_1a)$ and $e \notin \mathsf{row}(s_2a)$
10         $E = E \cup \mathsf{orb}(ae)$
11   Query $\mathcal{H} = \mathcal{A}(T)$ for equivalence
12   **if** the Teacher replies **no**, with a counter-example t
13         $E = E \cup \{\mathsf{orb}(t_0) \mid t_0 \text{ is a suffix of } t\}$
14   **until** the Teacher replies **yes** to the equivalence query.
15   **return** $\mathcal{H}$

We briefly recall the notation we use in the algorithm and afterwards prove convergence. We denote the observation table T by the pair $T = (S, E)$ of row and column indices. The membership will be queried for the set $SE \cup S\Sigma E$ and these observations will be stored during the algorithm. The function $\mathsf{row}_T \colon S \cup S\Sigma \to \mathcal{P}_{\mathsf{fs}}(E)$ returns the content of each row, i.e., $\mathsf{row}_T(t) \coloneqq \{e \in E \mid te \in \mathcal{L}\}$.[7] Since row_T takes values in a nominal join-semilattice, we use the notation $(\leq, <, \vee, \dots)$ from Section 4.1 on rows. Note that $\mathsf{row}_T(s) = s^{-1}\mathcal{L} \cap E$ can be thought of an approximation to $s^{-1}\mathcal{L}$. We will omit the subscript T in row.

Given an observation table $T = (S, E)$, we define the set of rows as

$$\mathsf{Rows}(T) \coloneqq \{\mathsf{row}(t) \mid t \in S \cup S\Sigma\}.$$

---

[7]It is more common to use 2^E instead of $\mathcal{P}_{\mathsf{fs}}(E)$ and use a more "functional" approach in which $\mathsf{row}_T(s)(e) = \mathcal{L}(se)$. However, we use $\mathcal{P}_{\mathsf{fs}}(E)$ to remain consistent with our notation in the paper.

This is an orbit-finite poset, ordered by $\leq$, that is, the order on $\mathcal{P}_{\mathsf{fs}}(E)$ given by subset inclusion. We define the set of *upper rows* as $\mathsf{Rows}^{\uparrow}(T) := \{\mathsf{row}(s) \mid s \in S\} \subseteq \mathsf{Rows}(T)$.

Definition 6.4. A table $T = (S, E)$ is
- *join-closed* if for each $s \in S$ and $a \in \Sigma$ we have
$$\mathsf{row}(sa) = \bigvee\{\mathsf{row}(s) \mid \mathsf{row}(s) \leq \mathsf{row}(sa), s \in \mathsf{JI}(\mathsf{Rows}(T)) \cap \mathsf{Rows}^{\uparrow}(T)\},$$
  in words: each extended row sa can be obtained as a join of join-irreducible rows in S;
- *join-consistent* if for all $s_1, s_2 \in S$ and $a \in \Sigma$ we have
$$\mathsf{row}(s_1) \leq \mathsf{row}(s_2) \quad \implies \quad \mathsf{row}(s_1a) \leq \mathsf{row}(s_2a).$$

Another way to define join-closedness would be to consider the set $\mathsf{JI}(\mathsf{Rows}^{\uparrow}(T))$ instead of $\mathsf{JI}(\mathsf{Rows}(T)) \cap \mathsf{Rows}^{\uparrow}(T)$. This would slightly change the algorithm, but not substantially. We stick to the original description of $\mathsf{NL}^{\star}$ [BHKL09].

Construction 6.5. Given a join-closed and join-consistent observation table $T = (S, E)$ we define an automaton $\mathcal{A}(T) = (\Sigma, Q, I, F, \delta)$ as follows:
$$Q = \mathsf{JI}(\mathsf{Rows}(T)) \cap \mathsf{Rows}^{\uparrow}(T)$$
$$I = \{r \in Q \mid r \leq \mathsf{row}(\epsilon)\}$$
$$F = \{r \in Q \mid \epsilon \in r\}$$
$$\delta = \{(\mathsf{row}(s), a, r') \mid s \in S, r' \in Q, r' \leq \mathsf{row}(sa)\}$$

This closely follows the definition of the canonical residual automaton in Theorem 4.10, but only uses the information from the observation table. This construction is effective, because one can decide whether $r \in \mathsf{Rows}(T)$ is a join-irreducible element if $r \neq \bigvee\{y \in \mathsf{Rows}(T) \mid y < x\}$ and r is non-empty. This is a set-builder expression in the programming language developed in [Boj19]. The rest of the construction is directly given as set-builder expressions.

In the remainder of this section we prove termination of our modified learning algorithm. In the following, $|X|$ is the *orbit-count* of an orbit-finite set X. By *atom-dimension* of X we mean the smallest k such that each element has a support of at most size k. Let $p(k)$ denote the number of orbits of the set $\mathbb{A}^{(k)} \times \mathbb{A}^{(k)}$; it equals the number of partial permutations on an k-element set.

Theorem 6.6. *Algorithm 1 query learns residual nominal languages. Moreover, it uses at most $\mathcal{O}(l+|\Sigma^{\leq l}|^2 \cdot p(dl))$ equivalence queries, where l is the length of the longest characterising word and d is the atom-dimension of Σ.*

The theorem will be proven with the help of the following lemmata.

Lemma 6.7. *Given an observation table (S, E), the algorithm extends it to a r-consistent table (S, E') in finitely many steps.*

Proof. Note that the row function defines a preorder $\sqsubseteq$ on S defined by $s_1 \sqsubseteq s_2$ iff $\mathsf{row}(s_1) \leq \mathsf{row}(s_2)$. Each time a column is added, this preorder is refined. So we obtain a chain of preorders:
$$\cdots \subsetneq \sqsubseteq_3 \subsetneq \sqsubseteq_2 \subsetneq \sqsubseteq_1 \subseteq S \times S.$$

Since the set $S \times S$ is orbit-finite, this chain has a length at most $|S \times S|$. The number of orbits in $S \times S$ is bounded by $|S|^2 \cdot p(k)$, where k is the atom-dimension of S.[8] $\qquad\square$

Lemma 6.8. *If the set S of an observation table $T = (S, E)$ contains all characterising words, then the table is join-closed.*

Proof. For this lemma, we consider the idealised observation table $T' = (S, \Sigma^*)$. In this table, we have $\mathsf{row}_{T'}(s) = s^{-1}\mathcal{L}$ for each $s \in S$. Since S contains all characterising words, we have $\mathsf{JI}(\mathsf{Rows}(T')) = \mathsf{JI}(\mathsf{Der}(\mathcal{L}))$. This means that the idealised table is join-closed, that is, for $s \in S$ and $a \in \Sigma$ we have

$$\mathsf{row}_{T'}(sa) = \bigvee_{s \in I} \mathsf{row}_{T'}(s),$$

for a suitable orbit-finite set of row indices I. This equation still holds if we restrict to the set E, and so the table T is join-closed. $\qquad\square$

Lemma 6.9. *Given an observation table (S, E) for a residual language, the algorithm extends it to a join-closed and join-consistent table (S', E') in finitely many steps.*

Proof. The algorithm adds Σ^l incrementally to the set S, with increasing l. In each step, we can make the table consistent (Lemma 6.7). And we only need to increase l until $\Sigma^{\leq l}$ contains all the required characterising words (Lemma 6.8). $\qquad\square$

Lemma 6.10. *The constructed hypothesis agrees with the values in the table.*

Proof. This closely follows the inductive proof of Theorem 4.10. Recall that the states of the automaton are given by $q = \mathsf{row}(s) \subseteq E$ for certain s. We prove $\mathcal{L}(\mathsf{row}(s)) \cap E = \mathsf{row}(s)$ for states $row(s)$ by induction:

$$
\begin{aligned}
\epsilon \in \mathcal{L}(\mathsf{row}(s)) \cap E &\iff \mathsf{row}(s) \in F \wedge \epsilon \in E \iff \epsilon \in \mathsf{row}(s) \wedge \epsilon \in E \\
au \in \mathcal{L}(\mathsf{row}(s)) \cap E &\iff u \in \mathcal{L}\left(\delta(\mathsf{row}(s), a)\right) \\
&\iff \exists s' \text{ with } \mathsf{row}(s') \leq \mathsf{row}(sa) \text{ and } u \in \mathcal{L}(\mathsf{row}(s')) \cap E \\
&\iff \exists s' \text{ with } \mathsf{row}(s') \leq \mathsf{row}(sa) \text{ and } u \in \mathsf{row}(s') \wedge u \in E \\
&\iff u \in \bigvee \delta(\mathsf{row}(s), a) \wedge u \in E \\
&\iff au \in \mathsf{row}(s) \wedge u \in E
\end{aligned}
$$

Note that we need E to be suffix-closed and that the empty word is in E. $\qquad\square$

We can now prove the main theorem of this section.

Proof of Theorem 6.6. Each time a counterexample is added, the next hypothesis (which will always be constructed per Lemma 6.9) will be different (Lemma 6.10). But this can only happen if a column or row is added, which we only need to do finitely many times (Lemmas 6.8 and 6.7). To be precise, this happens at most

$$l + |\Sigma^{\leq l}|^2 \cdot p(dl)$$

times, where p is from the proof of Lemma 6.7 and d is the atom-dimension of Σ and l is the least such that $\Sigma^{\leq l}$ contains a characterising word for each state of the canonical residual automaton. (Put differently: consider the shortest characterising words for all states, then l is the length of the longest of these.) So we conclude that the algorithm terminates after

---

[8]Slightly better bounds can be given by triangular numbers, but the asymptotics remain the same.

finitely many equivalence queries and only need finitely many membership queries since the table $(S \cup S\Sigma) \times E$ is orbit-finite. $\square$

Unfortunately, considering all words bounded by a certain length requires many membership queries. In fact, characterising words can be exponential in length [DLT02], meaning that this algorithm may need doubly exponentially many membership queries.[9]

Remark 6.11. Note that our termination argument is not concerned with the implementation of the teacher. This is standard for Angluin-style algorithms, which assume that the teacher is always able to provide correct answers to queries. As mentioned, this assumption is often too strong, and in our setting a direct equivalence check is not available due to Proposition 5.1. In practice, however, it is common to use testing techniques [Vaa17].

7. Discussion

7.1. **Conclusion.** In this paper we have investigated a subclass of nondeterministic automata over infinite alphabets. This class naturally arises in the context of query learning, where automata have to be constructed from finitely many observations. Although there are many classes of data languages, we have shown that our class of residual languages admit canonical automata. The states of these automata correspond to join-irreducible elements.

In the context of learning, we show that convergence of standard Angluin-style algorithms is not guaranteed, even for residual languages. We propose a modified algorithm which guarantees convergence at the expense of an increase in the number of observations.

We emphasise that, unlike other algorithms based on residuality such as $\mathsf{NL}^\star$ [BHKL09] and $\mathsf{AL}^\star$ [AEF15], our algorithm does not depend on the size, or even the existence, of the minimal deterministic automaton for the target language. This is a crucial difference, since dependence on the minimal deterministic automaton hinders generalisation to nondeterministic nominal automata, which are strictly more expressive. Ideally, in the residual case, one would like to have an efficient algorithm for which the complexity depends only on the length of characterising words, which is an intrinsic feature of residual automata. To the best of our knowledge, no such algorithm exists in the finite setting.

Finally, another interesting open question is whether all nondeterministic automata can be *efficiently* learned. We note that nondeterministic automata can be enumerated, and hence can be learned via equivalence queries only. This would result in a highly inefficient algorithm. This parallels the current understanding of learning probabilistic languages. Although efficient (learning in the limit) learning algorithms for deterministic and residual languages exist [DE04], the general case is still open.

7.2. **Related Work.**

---

[9]The reader should not interpret this as a complexity upper bound. In fact, no upper bound is known on the length of characterising words.

Lattices and Category Theory. In [GG17, GLP11] aspects of nominal lattices and Boolean algebras are investigated. To the best of our knowledge, our results of nominal lattice theory, especially the algorithmic properties (of join-irreducible elements), are new.

Residual automata over finite alphabets have categorical characterisation [MAMU15] in terms of *closure spaces*. We see no obstructions in generalising those results to nominal sets. This would amount to finding the right notion of nominal (complete) join-semilattice, with either finitely or uniformly supported joins, and the notion of nominal closure spaces.

Other Data Languages. Related data languages are the nominal languages with an explicit notion of binding [GC11, KMPS15, KST12, SKMW17]. Although these are sub-classes of the nominal languages we consider, binding is an important construct, e.g., to represent resource-allocation. Availability of a notion of derivatives [KMPS15] suggests that residuality may prove beneficial for learning these languages.

Another related type of automaton is that of session automata [BHLM14]. They differ in that they deal with global freshness as opposed to local freshness. They form a robust class of languages and their learnability is discussed in *loc. cit.*

Register automata can be defined parametrically in the data domain (see Section 2.3, but also [Boj19]). For the mathematical foundation of nominal sets (or sets with atoms), the only difference will be that of the group of symmetries. However, the current proof of Lemma 4.6 only works when the permutations are finite. This excludes the ordered atoms with monotone bijections as permutations. We do not know whether the main theorem still holds in the general case.

Alternating Automata. One could try to generalise $\mathsf{AL}^\star$ from [AEF15] to *alternating register automata*. Beside the join, these automata can also use a meet and so the algebraic structure will be that of a distributive (nominal) lattice.

We think that the analogue to Theorem 4.10 will hold: a language is accepted by a residual alternating nominal automaton iff there is a orbit-finite set of generators w.r.t. both the join and meet. However, there is no analogue to the join-irreducibles: there can be many different sets of generators. This might be an obstacle to generalise $\mathsf{AL}^\star$ as the algorithm will need to find a set of generating rows, and we are unsure whether this is even decidable.

Unambiguous Automata. Of special interest is the subclass of *unambiguous automata* which enjoy many recent breakthroughs [BC21, Col15, MQ19]. We note that residual languages are orthogonal to unambiguous languages. For instance, the language $\mathcal{L}_\mathrm{n}$ is unambiguous but not residual, whereas $\mathcal{L}_\mathrm{ng,r}$ is residual but ambiguous. Moreover, their intersection has neither property, and every deterministic language has both properties. One interesting fact is that if a canonical residual automaton is unambiguous, then the join-irreducibles form an anti-chain.

The unambiguous automata can be embedded into *weighted register automata* [BKM21]. This embedding allows one to use linear algebra for these automata and we expect that the learning algorithm for weighted automata [BM15, BV96] generalise to the setting of register automata. This could follow from the algebraic learning results of [US20], since the length of the nominal vector spaces is finite [BKM21]. The learning algorithm can therefore construct a minimal weighted register automaton, but this might not be an actual unambiguous (nondeterministic) automaton.

Acknowledgements

References

[AEF15]   Dana Angluin, Sarah Eisenstat, and Dana Fisman. Learning regular languages via alternating automata. In *IJCAI*, pages 3308–3314. AAAI Press, 2015.

[Ang87]   Dana Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.

[BBKL12] Mikołaj Bojańczyk, Laurent Braud, Bartek Klin, and Slawomir Lasota. Towards nominal computation. In *POPL*, pages 401–412, 2012.

[BC21]   Corentin Barloy and Lorenzo Clemente. Bidimensional linear recursive sequences and universality of unambiguous register automata. In *STACS*, volume 187 of *LIPIcs*, pages 8:1–8:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[BHKL09] Benedikt Bollig, Peter Habermehl, Carsten Kern, and Martin Leucker. Angluin-style learning of NFA. In *IJCAI*, pages 1004–1009, 2009.

[BHLM14] Benedikt Bollig, Peter Habermehl, Martin Leucker, and Benjamin Monmege. A robust class of data languages and an application to learning. *Logical Methods in Computer Science*, 10(4), 2014.

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

[BKM21]   Mikołaj Bojańczyk, Bartek Klin, and Joshua Moerman. Orbit-finite-dimensional vector spaces and weighted register automata. In *LICS*, 2021. To appear.

[BLLR17] Sebastian Berndt, Maciej Liśkiewicz, Matthias Lutter, and Rüdiger Reischuk. Learning residual alternating automata. In *AAAI*, pages 1749–1755, 2017.

[BM15]   Borja Balle and Mehryar Mohri. Learning weighted automata. In *CAI*, volume 9270 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 2015.

[Boj19]   Mikołaj Bojańczyk. *Slightly Infinite Sets*. Draft September 6, 2019, 2019.

[BT18]   Mikołaj Bojańczyk and Szymon Toruńczyk. On computability and tractability for infinite sets. In *LICS*, pages 145–154. ACM, 2018.

[BV96]   Francesco Bergadano and Stefano Varricchio. Learning behaviors of automata from multiplicity and equivalence queries. *SIAM J. Comput.*, 25(6):1268–1280, 1996.

[Col15]   Thomas Colcombet. Unambiguity in automata theory. In *Descriptional Complexity of Formal Systems - 17th International Workshop, DCFS 2015, Waterloo, ON, Canada, June 25-27, 2015. Proceedings*, pages 3–18, 2015.

[DE04]   François Denis and Yann Esposito. Learning classes of probabilistic automata. In *COLT*, volume 3120 of *Lecture Notes in Computer Science*, pages 124–139. Springer, 2004.

[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.

[DP02]   Brian A. Davey and Hilary A. Priestley. *Introduction to Lattices and Order, Second Edition*. Cambridge University Press, 2002.

[GC11]   Murdoch James Gabbay and Vincenzo Ciancia. Freshness and name-restriction in sets of traces with names. In *FOSSACSs*, pages 365–380, 2011.

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

[GG17]   Murdoch James Gabbay and Michael Gabbay. Representation and duality of the untyped λ-calculus in nominal lattice and topological semantics, with a proof of topological completeness. *Ann. Pure Appl. Logic*, 168(3):501–621, 2017.

[GLP11]   Murdoch James Gabbay, Tadeusz Litak, and Daniela Petrişan. Stone duality for nominal boolean algebras with N. In *CALCO*, volume 6859 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2011.

[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 *Lecture Notes in Computer Science*, pages 563–588. Springer, 2019.

[KF94]   Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.

[KMPS15]   Dexter Kozen, Konstantinos Mamouras, Daniela Petrişan, and Alexandra Silva. Nominal kleene coalgebra. In *ICAL*, pages 286–298, 2015.

[KS16]   Bartek Klin and Michał Szynwelski. SMT solving for functional programming over infinite structures. In *MSFP*, volume 207 of *EPTCS*, pages 57–75, 2016.

[KST12]   Alexander Kurz, Tomoyuki Suzuki, and Emilio Tuosto. On nominal regular languages with binders. In *FOSSACS*, pages 255–269, 2012.

[KT17]   Eryk Kopczynski and Szymon Toruńczyk. LOIS: syntax and semantics. In *POPL*, pages 586–598. ACM, 2017.

[KV94]   Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994. URL: `https://mitpress.mit.edu/books/introduction-computational-learning-theory`.

[KZ10]   Michael Kaminski and Daniel Zeitlin. Finite-memory automata with non-deterministic reassignment. *Int. J. Found. Comput. Sci.*, 21(5):741–760, 2010.

[MAMU15]   Robert S. R. Myers, Jirí Adámek, Stefan Milius, and Henning Urbat. Coalgebraic constructions of canonical nondeterministic automata. *Theor. Comput. Sci.*, 604:81–101, 2015.

[Moe19]   Joshua Moerman. *Nominal Techniques and Black Box Testing for Automata Learning*. PhD thesis, Radboud University, Nijmegen, The Netherlands, 2019.

[MQ19]   Antoine Mottet and Karin Quaas. The containment problem for unambiguous register automata. In *STACS*, pages 53:1–53:15, 2019.

[MS20]   Joshua Moerman and Matteo Sammartino. Residual nominal automata. In Igor Konnov and Laura Kovács, editors, *CONCUR*, volume 171 of *LIPIcs*, pages 44:1–44:21, 2020.

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

[Ner58]   Anil Nerode. Linear automaton transformations. *Proceedings of the AMS*, 9:541– 544, 1958.

[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.

[Pit13]   Andrew M. Pitts. *Nominal sets: Names and symmetry in computer science*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2013.

[SKMW17]   Lutz Schröder, Dexter Kozen, Stefan Milius, and Thorsten Wißmann. Nominal automata with name binding. In *FOSSACS*, pages 124–142, 2017.

[US20]   Henning Urbat and Lutz Schröder. Automata learning: An algebraic approach. In *LICS*, pages 900–914. ACM, 2020.

[Vaa17]   Frits W. Vaandrager. Model learning. *Commun. ACM*, 60(2):86–95, 2017.