Residual Nominal Automata

Joshua Moerman
RTWH Aachen University, Germany

Matteo Sammartino
Royal Holloway University of London, UK
University College London, UK

Abstract

We are motivated by the following question: which nominal 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 nominal automata*, a subclass of nondeterministic nominal 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 nominal 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.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects; Theory of computation → Automated reasoning

Keywords and phrases nominal automata, residual automata, derivative language, decidability, closure, exact learning, lattice theory

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2020.44

Related Version Full version at `https://arxiv.org/abs/1910.11666`.

Funding ERC AdG project 787914 FRAPPANT, EPSRC Standard Grant CLeVer (EP/S028641/1).

Acknowledgements We would like to thank Gerco van Heerdt for providing examples similar to that of $\mathcal{L}_r$ in the context of probabilistic automata. We thank Borja Balle for references on residual probabilistic languages, and Henning Urbat for discussions on nominal lattice theory. Lastly, we thank the reviewers of a previous version of this paper for their interesting questions and suggestions.

1 Introduction

Formal languages over infinite alphabets have received considerable attention recently. They include data languages for reasoning about XML databases [32], trace languages for analysis of programs with resource allocation [18], and behaviour of programs with data flows [19]. Typically, these languages are accepted by *register automata*, first introduced in the seminal paper [20]. Another appealing model is that of *nominal automata* [6]. While nominal automata are as expressive as register automata, they enjoy convenient properties. For example, the deterministic ones admit canonical minimal models, and the theory of formal languages and many textbook algorithms generalise smoothly.

In this paper, we investigate the properties of so-called *residual* nominal 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. [14] as a solution to the well-known problem of NFAs *not having* unique minimal representatives. They show that every regular language $\mathcal{L}$ admits a unique canonical RFSA.

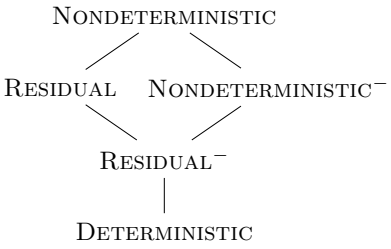


Figure 1 Relationship between classes of nominal languages. Edges are strict inclusions. With $\cdot^-$ we denote classes where automata are not allowed to *guess* values, i.e., to store symbols in registers without explicitly reading them.

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 [1] (deterministic automata are always residual), and enables efficient algorithms for learning nondeterministic [8] and alternating automata [2, 3]. Residuality has also been studied for learning probabilistic automata [13]. Existence of canonical residual automata is crucial for the convergence of these algorithms.

Our investigation of residuality in the nominal setting is motivated by the following question: which nominal languages admit an exact learning algorithm? In previous work [28], we have shown that the $\mathsf{L}^\star$ algorithm generalises smoothly to nominal languages, meaning that deterministic nominal automata can be learned. However, the general non-deterministic case proved to be significantly more challenging. In fact, in stark contrast with the finite-alphabet case, nondeterministic nominal automata are *strictly more expressive* than deterministic ones, thus 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 the nominal setting did not fully succeed: we could only prove that it works for deterministic languages, leaving the nondeterministic case open. By investigating residual languages, and how they relate to deterministic and nondeterministic ones, we are finally able to settle this case.

In summary, our contributions are as follows:

- Section 3: We refine nominal languages as depicted in Figure 1, by giving separating languages for each class.
- Section 4: We develop new results of nominal lattice theory, and we provide the main characterisation theorem (Theorem 4.10), showing that the class of residual languages allow for 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. Both properties are crucial for learning. We prove this important result by a machine-independent characterisation of those classes of languages. We also give an analogous result for non-guessing languages (Theorem 4.16).
- Section 5: We study decidability and closure properties. Many decision problems, such as equivalence and universality, are known to be undecidable for nondeterministic nominal automata. For residual automata, we show that universality becomes decidable. However, the problem of whether an automaton is residual is undecidable.

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

- Section 6: We settle important open questions about exact learning of nominal languages. We show that residuality does not imply convergence of existing algorithms, and we give a (modified) $\mathsf{NL}^\star$-style algorithm that works precisely for residual languages.

This research mirrors that of *residual probabilistic automata* [13]. 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.

2 Preliminaries

We recall the notions of nominal sets [33] and nominal automata [6]. Let $\mathbb{A}$ be a countably infinite set of *atoms*[2] 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 *nominal set* is a set X equipped with a function $\cdot\colon \mathsf{Perm}(\mathbb{A}) \times X \to X$, interpreting permutations over X. This function must be a *group action* of $\mathsf{Perm}(\mathbb{A})$, i.e., it must satisfy $\mathsf{id} \cdot x = x$ and $\pi \cdot (\pi' \cdot x) = (\pi \circ \pi') \cdot x$. We say that a set $A \subset \mathbb{A}$ *supports* $x \in X$ whenever $\pi \cdot x = x$ for all π fixing A, i.e., such that $\pi_{|A} = \mathsf{id}_A$. We require for nominal sets that each element x has a *finite* support. We denote by $\mathsf{supp}(x)$ the smallest finite set supporting x.

The *orbit* $\mathsf{orb}(x)$ of $x \in X$ is the set of elements in X reachable from x via permutations: $\mathsf{orb}(x) \coloneqq \{\pi \cdot x \mid \pi \in \mathsf{Perm}(\mathbb{A})\}$. X is *orbit-finite* whenever it is a finite union of orbits. Orbit-finite sets are finitely-representable, hence algorithmically tractable [5].

Given a nominal set X, a subset $Y \subseteq X$ is *equivariant* if it is preserved by permutations, i.e., $\pi \cdot Y = Y$, for all $\pi \in \mathsf{Perm}(\mathbb{A})$, where π acts element-wise. This definition extends to relations and functions. For instance, a function $f\colon X \to Y$ between nominal sets is equivariant whenever $\pi \cdot f(x) = f(\pi \cdot x)$. Given a nominal set X, the *nominal power set* is defined as $\mathcal{P}_{\mathsf{fs}}(X) \coloneqq \{U \subseteq X \mid U \text{ is finitely supported}\}$.

We recall the notion of nominal automaton from [6]. 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 2.1.** *A (nondeterministic)* nominal automaton $\mathcal{A}$ *consists of: an orbit-finite nominal set* Σ*, the alphabet; an orbit-finite nominal set of states* Q*; equivariant subsets* $I, F \subseteq Q$ *of initial and final states; and an equivariant subset* $\delta \subseteq Q \times \Sigma \times Q$ *of transitions.*

The usual notions of acceptance and language apply. We denote the language of $\mathcal{A}$ by $\mathcal{L}(\mathcal{A})$, and the language accepted by a state $q \in Q$ 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 $\mathsf{supp}(q)$.

We recall the notion of *derivative language* [14].[3]

▶ **Definition 2.2.** *Given a language* $\mathcal{L}$ *and a word* $u \in \Sigma^*$*, we define the* derivative *of* $\mathcal{L}$ *w.r.t.* u *as* $u^{-1}\mathcal{L} \coloneqq \{w \mid uw \in \mathcal{L}\}$ *and the set of all derivatives as* $\mathsf{Der}(\mathcal{L}) \coloneqq \left\{u^{-1}\mathcal{L} \mid u \in \Sigma^*\right\}$.

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

[2] Sometimes these are called *data values*.

[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.

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

▶ **Definition 2.3.** *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).

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

▶ **Theorem 2.4** ([6, 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 the sake of simplicity, we will 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 $\mathcal{L}_{\mathrm{d}} \coloneqq \{awa \mid a \in \mathbb{A}, w \in \mathbb{A}^*\}$. This is accepted by the following deterministic nominal automaton. The automaton is actually infinite-state, but we represent it symbolically using a register-like notation, where we annotate each state with the current

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

value of a register. Note that the derivatives $a^{-1}\mathcal{L}_{\mathrm{d}}, b^{-1}\mathcal{L}_{\mathrm{d}}, \ldots$ 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 $aa^{-1}\mathcal{L}_{\mathrm{d}}$.

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. Selecting the "right" derivatives is the key idea behind constructing residual automata in Theorem 4.10.

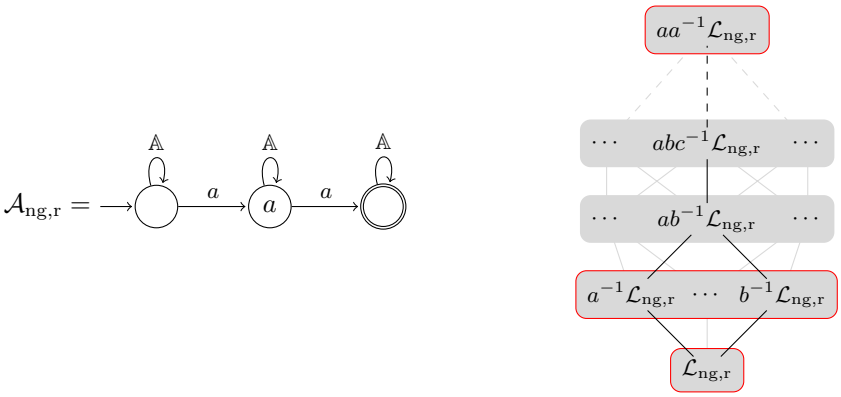


Figure 3 A (nonresidual) nondeterministic automaton accepting $\mathcal{L}_{\mathrm{ng,r}}$, and the poset $\mathsf{Der}(\mathcal{L}_{\mathrm{ng,r}})$.

Nondeterministic: Last letter is unique

The language is $\mathcal{L}_{\mathrm{n}} \coloneqq \{wa \mid a \text{ not in } w\} \cup \{\epsilon\}$. Derivatives $a^{-1}\mathcal{L}_{\mathrm{n}}$ are again unions of smaller languages: $a^{-1}\mathcal{L}_{\mathrm{n}} = \bigcup_{b \neq a} ab^{-1}\mathcal{L}_{\mathrm{n}}$. (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.) However, the poset $\mathsf{Der}(\mathcal{L})$ has an infinite descending chain of languages (with an increasing support), namely $a^{-1}\mathcal{L} \supset ab^{-1}\mathcal{L} \supset abc^{-1}\mathcal{L} \supset \ldots$ 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.

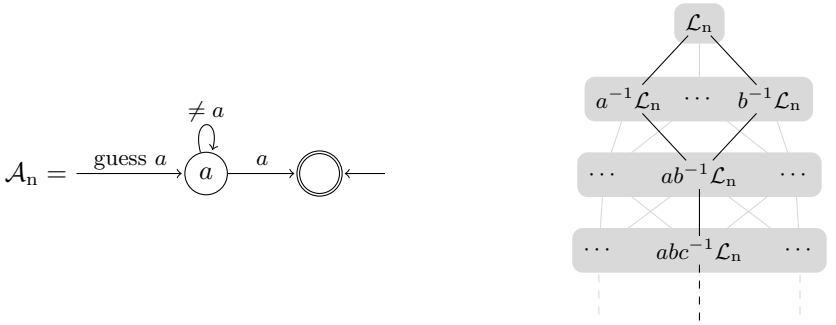


Figure 4 A nondeterministic automaton accepting $\mathcal{L}_{\mathrm{n}}$, and the poset $\mathsf{Der}(\mathcal{L}_{\mathrm{n}})$.

Residual: Last letter is unique but anchored

Consider the alphabet $\Sigma = \mathbb{A} \cup \{\mathsf{Anc}(a) \mid a \in \mathbb{A}\}$, where Anc is nothing more than a label. We add the transitions $(a, \mathsf{Anc}(a), a)$ to the automaton in the previous example. We obtain the language $\mathcal{L}_\mathrm{r} = \mathcal{L}(\mathcal{A}_\mathrm{r})$. 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 $\mathsf{Anc}(a)a^{-1}\mathcal{L}_\mathrm{r} = \{\epsilon\}$.

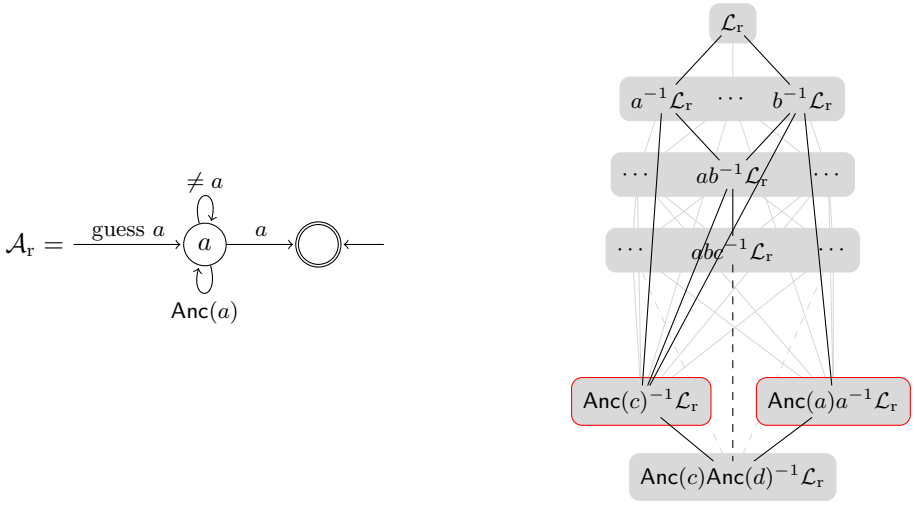


Figure 5 A residual automaton accepting $\mathcal{L}_\mathrm{r}$, and the poset $\mathsf{Der}(\mathcal{L}_\mathrm{r})$.

Non-guessing nondeterministic: Repeated atom with different successor

The language is $\mathcal{L}_\mathrm{ng} \coloneqq \{uabvac \mid u, v \in \mathbb{A}^*, a, b, c \in \mathbb{A}, b \neq a\}$. (We allow $a = b$ or $a = c$.) This is a language which can be accepted by a non-guessing automaton. 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.

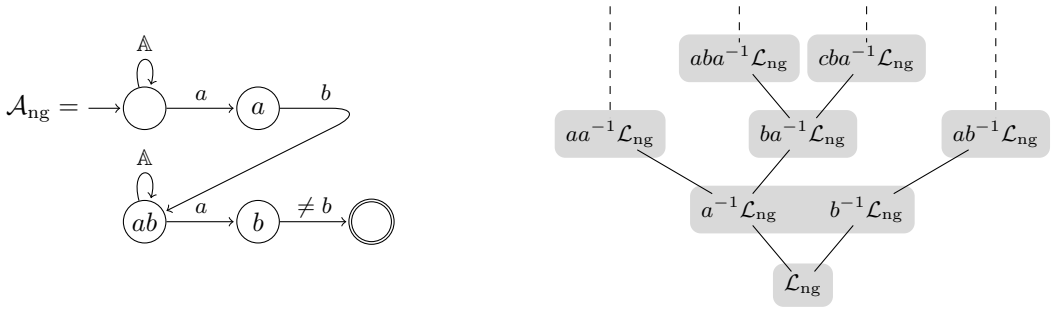


Figure 6 A deterministic automaton accepting $\mathcal{L}_\mathrm{ng}$, and the poset $\mathsf{Der}(\mathcal{L}_\mathrm{ng})$.

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 Nominal lattice theory

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 [17]. Moreover, the finitely supported union

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

is also equivariant. We note that this is more general than a binary union, but it is not a complete join semi-lattice. 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*

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

▶ 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 $x = \bigvee \mathfrak{x} \implies x \in \mathfrak{x}$, for every finitely supported $\mathfrak{x} \subseteq X$. The set of all join-irreducible elements is denoted by*

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

This is again an equivariant set.

▶ Remark 4.4. In lattice and order theory, join-irreducible elements are usually defined only for a lattice (see, e.g., [11]). 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 [8, 14, 28]. 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 [11, 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 a join-irreducible $x \in X$ such that $x \leq a$ and $x \not\leq b$.*
2. *Let $a \in X$, then $a = \bigvee \{x \in X \mid x \text{ join-irreducible in } X \text{ and } x \leq a\}$.*

[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.

▶ **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 $\mathsf{JI}(Y) = \mathsf{JI}(X)$.*

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 ⇒ 2) Take the set of languages accepted by the states: $J \coloneqq \{\mathcal{L}(q) \mid q \in \mathcal{A}\}$. 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 ⇒ 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 ⇒ 1) We can construct the following residual automaton, whose language is exactly $\mathcal{L}$:

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

First, note that $\mathcal{A} \coloneqq (\Sigma, Q, I, F, \delta)$ is a well-defined nominal automaton. In fact, all the components are orbit-finite, and equivariance of $\leq$ implies equivariance of δ. Second, we show by induction on words that each state $q = w^{-1}\mathcal{L}$ accepts its corresponding language, namely $\mathcal{L}(q) = w^{-1}\mathcal{L}$.

$$\begin{aligned} \epsilon \in \mathcal{L}(w^{-1}\mathcal{L}) &\iff w^{-1}\mathcal{L} \in F \iff \epsilon \in w^{-1}\mathcal{L} \\ au \in \mathcal{L}(w^{-1}\mathcal{L}) &\iff u \in \mathcal{L}\left(\delta(w^{-1}\mathcal{L}, a)\right) \\ &\iff u \in \mathcal{L}\left(\left\{v^{-1}\mathcal{L} \in Q \mid v^{-1}\mathcal{L} \leq wa^{-1}\mathcal{L}\right\}\right) \\ &\overset{\text{(i)}}{\iff} u \in \bigvee \left\{v^{-1}\mathcal{L} \in Q \mid v^{-1}\mathcal{L} \leq wa^{-1}\mathcal{L}\right\} \\ &\iff \exists v^{-1}\mathcal{L} \in Q \text{ with } v^{-1}\mathcal{L} \leq wa^{-1}\mathcal{L} \text{ and } u \in v^{-1}\mathcal{L} \\ &\overset{\text{(ii)}}{\iff} u \in wa^{-1}\mathcal{L} \iff au \in w^{-1}\mathcal{L} \end{aligned}$$

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 $\mathcal{L}$. ◀

▶ **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.*

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 $\mathcal{L} \coloneqq \{abb\dots b \mid a \neq b\}$ 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:

$$\epsilon^{-1}\mathcal{L} = \{abb\dots b \mid a \neq b\}$$
$$ab^{-1}\mathcal{L} = \{bb\dots b\} \qquad (a, b \in \mathbb{A} \text{ distinct})$$

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, $a^{-1}\mathcal{L}_{\mathrm{n}} = \bigcup_{b\neq a} ab^{-1}\mathcal{L}_{\mathrm{n}}$. This means that $\mathsf{JI}(\mathsf{Der}(\mathcal{L}_{\mathrm{n}})) = \emptyset$, hence it does not generate $\mathsf{Der}(\mathcal{L}_{\mathrm{n}})$ and by Theorem 4.10 there is no residual automaton.

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

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

This in itself is not a problem: the language $\mathcal{L}_{\mathrm{ng,r}}$ also has an infinite ascending chain. However, for $\mathcal{L}_{\mathrm{ng}}$, none of the languages in this chain are a union of smaller derivatives. Put differently: all the languages in this chain are join-irreducible (see appendix for the details). So the set $\mathsf{JI}(\mathsf{Der}(\mathcal{L}_{\mathrm{ng}}))$ is *not orbit-finite*. By Theorem 4.10, we conclude that there is no residual automaton accepting $\mathcal{L}_{\mathrm{ng}}$.

▶ Remark 4.14. 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 $Y \subseteq \mathcal{P}_{\mathsf{fs}}(\Sigma^*)$ 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 [20] does not allow for guessing, and non-guessing automata remain actively researched [29]. Register automata with guessing were introduced in [21], 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 4.15.** *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 4.16.** *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.

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

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. ◀

To better understand the structure of the canonical non-guessing residual automaton, we recall the following fact (see [33] for details) and its consequence on non-guessing automata.

▶ **Lemma 4.17.** *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 4.18.** *The transition relation δ of non-guessing automata can be equivalently be described as a function $\delta \colon Q \times \Sigma \to \mathcal{P}_{\mathrm{fin}}(Q)$, where $\mathcal{P}_{\mathrm{fin}}(Q)$ 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 4.16, instead of uniformly supported unions.

5 Decidability and Closure Results

In this section we investigate decidability and closure properties. 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 [4].

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

Second, a negative result: determining whether an automaton is 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, thus "getting around" this undecidability issue.

▶ **Proposition 5.2.** *The problem of determining whether a given nondeterministic nominal automaton is residual is undecidable.*

The above result is obtained by reducing the universality problem for general nondeterministic nominal automata to the residuality problem. Given an automaton $\mathcal{A}$, we construct another automaton $\mathcal{A}'$ which is residual if and only if $\mathcal{A}$ is universal (see appendix for details). This result also holds for the subclass of non-guessing automata, as the construction of $\mathcal{A}'$ does not introduce any guessing and universality for non-guessing nondeterministic nominal automata is undecidable.

▶ Remark 5.3. Equivalence between residual nominal automata is still an open problem. The usual proof of undecidability of equivalence is via a reduction from universality. This proof does not work anymore, because universality for residual automata is decidable (Proposition 5.1). We conjecture that equivalence remains undecidable for residual automata.

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 [6].

Closure under Kleene star is yet to be settled.

6 Exact learning

In our previous paper on learning nominal automata [28], we provided an exact learning algorithm for nominal deterministic languages. Moreover, we observed by experimentations that the algorithm was also able to learn specific nondeterministic languages. However, several questions on nominal languages remained open, most importantly:

- Which nominal languages can be characterised via a finite set of observations?
- Which nominal languages admit an Angluin-style learning algorithm?

In this section we will answer these questions using the theory developed in the previous sections.

6.1 Angluin-style learning

We briefly review the classical automata learning algorithms $\mathsf{L}^\star$ by Angluin [1] for deterministic automata, and $\mathsf{NL}^\star$ by Bollig et al. [8] for residual automata. Then we discuss convergence in the nominal setting.

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, i.e., $\mathcal{L} = \mathcal{L}(\mathcal{H})$?

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 [22] or [35] for details).

Observations about the language made by the learner via queries are stored in an *observation table* $\mathcal{T}$. This is a table where rows and columns range over two finite sets of words $S, E \subseteq \Sigma^\star$ respectively, and $\mathcal{T}(u, v) = 1$ if and only if $uv \in \mathcal{L}$. Intuitively, each row of $\mathcal{T}$ approximates a derivative of $\mathcal{L}$, in fact we have $\mathcal{T}(u) \subseteq u^{-1}\mathcal{L}$. However, the information contained in $\mathcal{T}$ 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 $\mathcal{T}(u)$, $\mathcal{T}(v)$ 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 $\mathcal{T}$ is *closed* whenever one-letter extensions of derivatives are already in the table, i.e., $\mathcal{T}$ has a row for $ua^{-1}\mathcal{L}$, for all $u \in S, a \in \Sigma$. If the table is closed,[5] $\mathsf{L}^\star$ is able to construct an automaton from $\mathcal{T}$, where states are distinct rows (i.e., derivatives). The construction follows the classical one for the canonical automaton of a language from its derivatives [31]. The $\mathsf{NL}^\star$ 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 $\mathcal{T}$), and hence is able to learn a RFSA accepting the target language.

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

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[6], after which the process is repeated iteratively.

6.2 The nominal case

In [28] 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. The algorithm $\nu\mathsf{L}^\star$ always terminates for deterministic languages, because distinct derivatives, and hence distinct rows in the observation table, are orbit-finitely many (see Theorem 2.4).

However, it will *never* terminate for languages not accepted by deterministic automata (such as residual or nondeterministic languages).

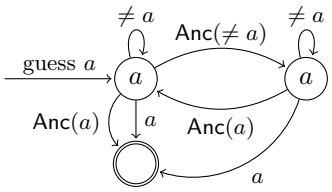
▶ **Theorem 6.1** ([27]). *$\nu\mathsf{L}^\star$ converges if and only if $\mathsf{Der}(\mathcal{L})$ is orbit-finite, in which case it outputs the canonical deterministic automaton accepting $\mathcal{L}$. Moreover, at most $\mathcal{O}(nk)$ equivalence queries are needed, where n is the number of orbits 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. Then $\mathcal{L}$ can be represented via an observation table with orbit-finitely-many rows and columns if and only if $\mathcal{L}$ is residual. Rows of this table correspond to join-irreducible derivatives.*

This explains why in [28] $\nu\mathsf{NL}^\star$ 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 current $\nu\mathsf{NL}^\star$ algorithm does not guarantee that it 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

[6] $\mathsf{L}^\star$ and $\mathsf{NL}^\star$ adopt different counterexample-handling strategies: the former adds a new row, the latter a new column. Both result in a new derivative being detected.

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.

While there is no hope of convergence in the non-residual case, as no orbit-finite observation table exists characterising derivatives, we now propose a modification of $\nu\mathsf{NL}^\star$ which guarantees termination for residual languages.

▶ **Theorem 6.4.** *There is an algorithm which query learns residual nominal languages.*

Proof (Sketch). When the algorithm adds a word w to the set of rows, then it also adds all other words of length $|w|$.[7] Since all words of bounded length are added, the algorithm will eventually find all words that are characterising for states of the canonical residual automaton, and it will therefore be able to reconstruct this automaton. See appendix for details. ◀

Unfortunately, considering all words bounded by a certain length requires many membership queries. In fact, characterising words can be exponential in length [14], meaning that this algorithm may need doubly exponentially many membership queries.

▶ Remark 6.5. 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 [12], the general case is still open.

7 Conclusions, related and future work

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 $\mathsf{NL}^\star$ [8] and $\mathsf{AL}^\star$ [2], 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 an 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.

[7] The set $\{w \in \Sigma^* \mid |w| = k\}$ is orbit-finite, for any fixed $k \in \mathbb{N}$.

We also show that universality is decidable for residual automata, in contrast to undecidability in the general nondeterministic case. As future work, we plan to attack the language inclusion/equivalence problem for residual automata. This is a well-known and challenging problem for data languages, which has been answered for specific subclasses [9, 10, 29, 34].

Of special interest is the subclass of *unambiguous automata* [10, 29]. 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.

Other related work are nominal languages/expressions with an explicit notion of binding [15, 25, 26, 34]. Although these are sub-classes of nominal languages, binding is an important construct, e.g., to represent resource-allocation. Availability of a notion of derivatives [25] suggests that residuality may prove beneficial for learning these languages.

Residual automata over finite alphabets also have a categorical characterisation [30]. 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.

Finally, in [16, 17] aspects of nominal lattices and Boolean algebras are investigated. To the best of our knowledge, our results of nominal lattice theory, especially those on join-irreducibles, are new.

References

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

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

3 Sebastian Berndt, Maciej Liśkiewicz, Matthias Lutter, and Rüdiger Reischuk. Learning residual alternating automata. In *AAAI*, pages 1749–1755, 2017. URL: `http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14748`.

4 Mikołaj Bojańczyk. *Slightly Infinite Sets*. Draft September 6, 2019, 2019. URL: `https://www.mimuw.edu.pl/~bojan/paper/atom-book`.

5 Mikołaj Bojańczyk, Laurent Braud, Bartek Klin, and Slawomir Lasota. Towards nominal computation. In *POPL*, pages 401–412, 2012. `doi:10.1145/2103656.2103704`.

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

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

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

9 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. `doi:10.2168/LMCS-10(4:19)2014`.

10 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. `doi:10.1007/978-3-319-19225-3_1`.

11 Brian A. Davey and Hilary A. Priestley. *Introduction to Lattices and Order, Second Edition*. Cambridge University Press, 2002. `doi:10.1017/CBO9780511809088`.

12 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.

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

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

15 Murdoch James Gabbay and Vincenzo Ciancia. Freshness and name-restriction in sets of traces with names. In *FOSSACSs*, pages 365–380, 2011. doi:10.1007/978-3-642-19805-2_25.

16 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. doi:10.1016/j.apal.2016.10.001.

17 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.

18 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.

19 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.

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

21 Michael Kaminski and Daniel Zeitlin. Finite-memory automata with non-deterministic reassignment. *Int. J. Found. Comput. Sci.*, 21(5):741–760, 2010. doi:10.1142/S0129054110007532.

22 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.

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

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

25 Dexter Kozen, Konstantinos Mamouras, Daniela Petrişan, and Alexandra Silva. Nominal kleene coalgebra. In *ICAL*, pages 286–298, 2015. doi:10.1007/978-3-662-47666-6_23.

26 Alexander Kurz, Tomoyuki Suzuki, and Emilio Tuosto. On nominal regular languages with binders. In *FOSSACS*, pages 255–269, 2012. doi:10.1007/978-3-642-28729-9_17.

27 Joshua Moerman. *Nominal Techniques and Black Box Testing for Automata Learning*. PhD thesis, Radboud University, Nijmegen, The Netherlands, 2019. URL: http://hdl.handle.net/2066/204194.

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

29 Antoine Mottet and Karin Quaas. The containment problem for unambiguous register automata. In *STACS*, pages 53:1–53:15, 2019. doi:10.4230/LIPIcs.STACS.2019.53.

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

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

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

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

34 Lutz Schröder, Dexter Kozen, Stefan Milius, and Thorsten Wißmann. Nominal automata with name binding. In *FOSSACS*, pages 124–142, 2017. doi:10.1007/978-3-662-54458-7_8.

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

A Omitted proofs

▶ 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 $Y \subseteq \langle X \rangle$.

Proof. Take any $\mathfrak{x} \subseteq \langle X \rangle$ finitely supported. All $x \in \mathfrak{x}$ are of the form $\bigvee \mathfrak{y}_x$, for some $\mathfrak{y}_x \subseteq X$ finitely supported. Consider the finitely supported set $T = \{y \mid y \in \mathfrak{y}_x, x \in \mathfrak{x}\} \subseteq X$. Then we see that $\bigvee \mathfrak{x} = \bigvee T \in \langle X \rangle$, meaning that $\langle X \rangle$ is closed under $\bigvee$. The second part of the claim is easy: any set closed under $\bigvee$ and containing X must also contain $\langle X \rangle$. ◀

▶ **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 a join-irreducible $x \in X$ 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. In this proof we need a technicality. Let P be a finitely supported, non-empty poset (i.e., both P and $\leq$ are supported by a finite $A \subset \mathbb{A}$). If P is A-orbit-finite then P has a minimal element, as we can consider the finite poset of A-orbits and find a minimal A-orbit. Here we use the notion of an A-orbit, i.e., an orbit defined over permutations that fix A. (See [33, Chapter 5] for details.)

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 $m \in S$. 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. ◀

▶ **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 $X \subseteq \langle \mathsf{JI}(Y_0) \rangle$). 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 $x_0 \in \mathfrak{x}$. ◀

▶ **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 it is *saturated*, meaning that no transitions can be added without changing the language. ◀

▶ **Example 4.13.** All the languages in the following ascending chain are join-irreducible.

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

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

Observe that if u is a suffix of w, then $u^{-1}\mathcal{L}_{\mathrm{ng}} \subseteq w^{-1}\mathcal{L}_{\mathrm{ng}}$. 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 $aaa_0 \in u^{-1}\mathcal{L}_{\mathrm{ng}}$, but $aaa_0 \notin w^{-1}\mathcal{L}_{\mathrm{ng}}$. If a is succeeded by b (not necessarily distinct), then either aa or aa_0 is in $u^{-1}\mathcal{L}_{\mathrm{ng}}$. But neither aa nor aa_0 is in $w^{-1}\mathcal{L}_{\mathrm{ng}}$. This shows that for $u^{-1}\mathcal{L}_{\mathrm{ng}} \subseteq w^{-1}\mathcal{L}_{\mathrm{ng}}$, we necessarily have that $u \in \{a_0, \dots, a_k\}^*$. (This also means that automatically $\mathsf{supp}(u^{-1}\mathcal{L}_{\mathrm{ng}}) \subseteq \mathsf{supp}(w^{-1}\mathcal{L}_{\mathrm{ng}})$.)

Second, when $u = \epsilon$, we have $u^{-1}\mathcal{L}_{\mathrm{ng}} \subseteq w^{-1}\mathcal{L}_{\mathrm{ng}}$. And for $|u| = 1$, if $u = a_0$, then $u^{-1}\mathcal{L}_{\mathrm{ng}} \subseteq w^{-1}\mathcal{L}_{\mathrm{ng}}$. If $u = a_i$ with $i > 0$, then $a_i a_i a_{i-1} \in u^{-1}\mathcal{L}_{\mathrm{ng}}$, but that word is not in $w^{-1}\mathcal{L}_{\mathrm{ng}}$. This shows that for $u^{-1}\mathcal{L}_{\mathrm{ng}} \subseteq w^{-1}\mathcal{L}_{\mathrm{ng}}$ with $|u| \leq 1$, we necessarily have that u is a suffix of w.

Third, we prove the same for $|u| \leq 2$. 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 $u^{-1}\mathcal{L}_{\mathrm{ng}}$, but not in $w^{-1}\mathcal{L}_{\mathrm{ng}}$. Suppose that u contains $a_0 a_i$ $(i > 0)$ or $a_0 a_0$, then $u^{-1}\mathcal{L}_{\mathrm{ng}}$ contains either $a_0 a_0$ or $a_0 a_1$ respectively. Neither of these words are in $w^{-1}\mathcal{L}_{\mathrm{ng}}$. This shows that $u^{-1}\mathcal{L}_{\mathrm{ng}} \subseteq w^{-1}\mathcal{L}_{\mathrm{ng}}$ 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 $u^{-1}\mathcal{L}_{\mathrm{ng}} \subseteq w^{-1}\mathcal{L}_{\mathrm{ng}}$.

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 $u^{-1}\mathcal{L}_{\mathrm{ng}}$, but not in $w^{-1}\mathcal{L}_{\mathrm{ng}}$. This shows that u has to end in $a_1 a_0$. That is, for $u^{-1}\mathcal{L}_{\mathrm{ng}} \subseteq w^{-1}\mathcal{L}_{\mathrm{ng}}$ with $|u| \geq 2$, we necessarily have that u is a suffix of w.

So far, we have shown that

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

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

▶ **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. 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 [7]) and implementations exist to compute with atoms [23, 24].

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. ◀

▶ **Proposition 5.2.** *The problem of determining whether a given nondeterministic nominal automaton is residual is undecidable.*

Proof. The construction is inspired by [14, Proposition 8.4].[8] We show undecidability by reducing the universality problem for nominal automata to the residuality problem.

Let $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ be a nominal (nondeterministic) automaton on the alphabet Σ. We first extend the alphabet:

$$\Sigma' = \Sigma \cup \{\underline{q} \mid q \in Q\} \cup \{\overline{q} \mid q \in Q\} \cup \{\$, \#\},$$

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

$$\begin{aligned}
Q' &= \{\overline{q} \mid q \in Q\} \cup \{\underline{q} \mid q \in Q\} \cup \{\top, x, y\} \\
I' &= \{\underline{q} \mid q \in Q\} \cup \{x, y\} \\
F' &= \{\overline{q} \mid q \in F\} \cup \{\top\} \\
\delta' &= \{(\overline{q}, a, \overline{q}' \mid (q, a, q') \in \delta\} \cup \{(\underline{q}, \underline{q}, \underline{q}) \mid q \in Q\} \cup \{(\underline{q}, \overline{q}, \overline{q}) \mid q \in Q\} \\
&\quad \cup \{(x, \$, \top), (x, \#, x), (y, \#, y)\} \cup \{(\top, a, \top) \mid a \in \Sigma\} \cup \{(y, \$, \overline{i}) \mid i \in I\}
\end{aligned}$$

See Figure 7 for a sketch of the automaton $\mathcal{A}'$. The blue part is a copy of the original automaton. The red part forces the original states to be residual, by providing anchors to each state. Finally the orange part is the interesting part. 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. For the original states of $\mathcal{A}$ the property holds, as we can provide anchors: All the states $\overline{q}$ and $\underline{q}$ are anchored by the words $\overline{q}$ and $\underline{q}$ respectively. Then we consider the states x and $\top$, their languages are $\mathcal{L}(\top) = \Sigma^* = \$^{-1}\mathcal{L}(\mathcal{A}')$ 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.

[8] 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.

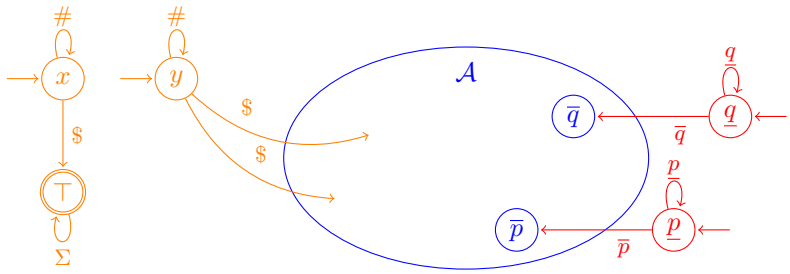


Figure 7 Sketch of the automaton $\mathcal{A}'$ constructed in the proof of Proposition 5.2.

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 $a \in \Sigma$, $\overline{q}$, $\underline{q}$ 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. Moreover, the construction $\mathcal{A} \mapsto \mathcal{A}'$ is effective, as it performs computations with orbit-finite sets. ◀

▶ **Theorem 6.4.** *There is an algorithm which query learns residual nominal languages.*

Proof. As explained in the text, we modify the $\nu\mathsf{NL}^\star$ algorithm from [28]: When the table is not closed, we not only add the missing words, but all the words of the same length. This guarantees that the algorithm finds rows for all join-irreducible derivatives, i.e., all states of the canonical residual automaton.

The pseudocode is given in Algorithm 1, where the modifications to $\nu\mathsf{NL}^\star$ are highlighted in red. We briefly explain the notation. An observation table $\mathcal{T}$ is defined by a set of row (resp. column) indices S (resp. E). The value $\mathcal{T}(s, e)$ is given by $\mathcal{L}(se)$ (we may do this via membership queries). We denote the set of rows by $\mathrm{Rows}(S, E) \coloneqq \{\mathrm{row}(s) \mid s \in S\Sigma \cup S\}$,

Algorithm 1 Modified nominal $\mathsf{NL}^\star$ algorithm for Theorem 6.4.

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

$S, E = \{\epsilon\}$
repeat
  while (S, E) is not residually-closed or not residually-consistent
  if (S, E) is not residually-closed
    find $s \in S, a \in A$ such that $row(sa) \in \mathsf{JI}(\mathrm{Rows}(S, E)) \setminus \mathrm{Rows}^\top(S, E)$
    $k =$ length of the word sa
    $S = S \cup \Sigma^{\leq k}$
  if (S, E) is not residually-consistent
    find $s_1, s_2 \in S$, $a \in A$, and $e \in E$ such that $\mathrm{row}(s_1) \sqsubseteq \mathrm{row}(s_2)$ and $\mathcal{L}(s_1ae) = 1$, $\mathcal{L}(s_2ae) = 0$
    $E = E \cup \mathsf{orb}(ae)$
  Make the conjecture $N(S, E)$
  if the Teacher replies **no**, with a counter-example t
    $E = E \cup \{\mathsf{orb}(t_0) \mid t_0$ is a suffix of $t\}$
until the Teacher replies **yes** to the conjecture $N(S, E)$.
return $N(S, E)$

where $\mathrm{row}(s)(e) = \mathcal{T}(s, e)$. Note that $\mathrm{Rows}(S, E)$ also includes rows for one-letter extensions. The set of rows labelled by S is denoted by $\mathrm{Rows}^{\top}(S, E) \coloneqq \{\mathrm{row}(s) \mid s \in S\}$. The set $\mathrm{Rows}(S, E)$ is a poset, ordered by $r_1 \sqsubseteq r_2$ iff $r_1(e) \leq r_2(e)$ for all $e \in E$. To construct a hypothesis $N(S, E)$, we use the construction from Theorem 4.10, where $\mathrm{Rows}(S, E)$ plays the role of $\mathsf{Der}(\mathcal{L})$.

We can give a bound to the number of equivalence queries. Given an orbit-finite nominal set X, let $|X|$ be the number of its orbit. Then equivalence queries are bounded by $\mathcal{O}(m + |\Sigma^{\leq m+1}| \times k)$, where m is the length of the longest characterising word and k is the maximum support size of the canonical residual automaton. Intuitively, each of the rows in the table could be a separate state, and for each state there is some work to be done, concerning learning the right support and local symmetries (see [27] for details on this). ◀