Residual Nominal Automata[⋆]

Joshua Moerman[1] and Matteo Sammartino[2]

[1] RTWH Aachen University, joshua@cs.rwth-aachen.de
[2] University College London, m.sammartino@ucl.ac.uk

Abstract. Nominal automata are models for accepting languages over infinite alphabets. In this paper we refine the hierarchy of nondeterministic nominal automata, by developing the theory of *residual automata.* In particular, we show that they admit canonical minimal representatives, and the universality problem becomes decidable. We also study exact learning of these automata, and settle questions that were left open about their learnability via observations. Finally, we unveil connections with probabilistic automata.

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 [19], and behaviour of programs with data flows [20]. Typically, these languages are accepted by *register automata*, first introduced in the seminal paper [21]. 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 sub-class of nondeterministic finite automata (NFAs) introduced by Denis et al. [15] 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.

Obtaining an analogous result for nominal languages is significantly more challenging. In fact, in stark contrast with the finite-alphabet case, nondeterministic nominal automata are *strictly more expressive* than deterministic ones. Therefore, residual automata are not just succinct representations of deterministic languages. Indeed, we shall see that the situation is the one depicted in Fig. 1. With $\cdot^{-}$ we denote classes where automata are not allowed to *guess*, that is, to store symbols in registers without explicitly reading them. The original

[⋆] This research has been partially funded by the ERC AdG project 787914 FRAPPANT and a Leverhulme Prize (PLP-2016-129).

$$\begin{tikzcd}
 & \textsc{Nondeterministic} \arrow[dl, no head] \arrow[dr, no head] & \\
\textsc{Residual} \arrow[dr, no head] & & \textsc{Nondeterministic}^{-} \arrow[dl, no head] \\
 & \textsc{Residual}^{-} \arrow[d, no head] & \\
 & \textsc{Deterministic} &
\end{tikzcd}$$

Fig. 1. Relationship between nominal languages. Edges are strict inclusions.

definition of register automaton in [21] is without guessing. Albeit this definition is more natural, it has a few issues. For instance, the languages accepted by non-guessing automata are not closed under reversal. Register automata with guessing were introduced in [22].

In this paper we will show that each inclusion in Fig. 1 is *strict*. More importantly, the deterministic and residual classes allow for canonical automata which are minimal in their respective classes and unique (up to isomorphism). We shall prove this important result by a machine-independent characterisation of those classes of languages. Many decision problems, such as equivalence and universality, are known to be undecidable for nominal automata. For residual automata, we will show that universality becomes decidable. However, the problem of whether an automaton is residual is undecidable.

Residual automata play a key role in the context of *exact learning*[3], 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 enables efficient algorithms for learning nondeterministic [8] and alternating automata [2,4]. Residuality has also been studied for learning probabilistic automata [14]. The existence of canonical automata is crucial for the convergence of these algorithms. In a previous paper [29], we have shown that it is possible to exactly learn automata for deterministic nominal languages. It was left open whether nondeterministic nominal automata can be efficiently learned. Motivated by the existence of canonical residual nominal automata, in this paper we investigate and settle this question.

This research mirrors that of *residual probabilistic automata* [14]. There, too, one has distinct classes of which the deterministic and residual ones admit canonical automata and have an algebraic characterisations. We believe that our results contribute to a better understanding of learnability of automata with some sort of nondeterminism.

Outline of the paper. After recalling preliminaries of nominal set theory in Section 2, in Section 3 we give languages that separate the various classes as depicted in Fig. 1. In Section 4 we introduce notions of nominal lattice theory, and we provide the main characterisation theorem (Theorem 2). We show an analogous result for the non-guessing case (Theorem 3). In Section 5 we show

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

that universality is decidable for residual automata, but residuality is not. Finally, in Section 6, we settle important open questions about exact learning of nominal languages, and we establish connections with probabilistic automata.

2 Preliminaries

We recall the notions of nominal sets [33] and nominal automata [6]. Let $\mathbb{A}$ be a countably infinite set of *atoms*[4] 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, for all π acting as the identity on A, we have $\pi \cdot x = x$. We require for nominal sets that each element x has a *finite* support. The least support of $x \in X$, denoted $\mathsf{supp}(x)$, is 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, explicitly

$$\mathsf{orb}(x) \coloneqq \{\pi \cdot x \mid \pi \in \mathsf{Perm}(\mathbb{A})\}\,.$$

Then X is *orbit-finite* whenever it is a finite union of orbits.

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 elementwise. 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

$$\mathcal{P}(X) \coloneqq \{U \subseteq X \mid U \text{ is finitely supported}\}\,.$$

We recall the notion of nominal automaton from [6]. The theory of nominal automata seamlessly extend classical automata theory by having orbit-finite nominal sets and equivariant functions in place of finite sets and functions.

Definition 1. *A (non-deterministic)* nominal automaton $\mathcal{A}$ *consists of:*

- *an orbit-finite nominal set Σ, the alphabet,*
- *an orbit-finite nominal set of states Q,*
- *an equivariant subsets $I, F \subseteq Q$ of initial and final states,*
- *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 $\mathcal{L}(\mathcal{A}) \in \mathcal{P}(\Sigma^*)$ is equivariant, and that $\mathcal{L}(q) \in \mathcal{P}(\Sigma^*)$ need not be equivariant, but it is supported by $\mathsf{supp}(q)$.

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

We recall the notion of *derivative language* [15].[5]

Definition 2. *Given a language $\mathcal{L} \in \mathcal{P}(\Sigma^*)$ and a word $w \in \Sigma^*$, the* derivative *of $\mathcal{L}$ w.r.t. w is*

$$w^{-1}\mathcal{L} \coloneqq \{u \mid wu \in \mathcal{L}\}\,.$$

The set of all derivatives of $\mathcal{L}$ is:

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

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 automata and non-guessing nominal automata, which we introduce next.

Definition 3. *Given a nominal automaton $\mathcal{A}$.*

- *It is* deterministic *if 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 $\delta\colon Q \times \Sigma \to Q$.*
- *It is is* 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.*
- *It is* non-guessing *if, for each $(q, a, q') \in \delta$, $\mathsf{supp}(q') \subseteq \mathsf{supp}(q) \cup \mathsf{supp}(a)$.*

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 1 ([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.*

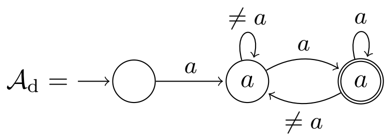
3 Separating languages

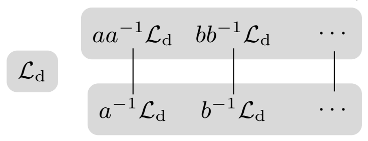
Deterministic, non-deterministic and residual automata have the same expressive power when dealing with finite alphabets. The situation is more nuanced in the nominal setting. We now give one language for each of the classes in Figure 1. These languages separate the different classes, meaning that they belong to the respective class, but not to the classes below (or beside) it.

[5] 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.

For each example, we depict an automaton recognising the language (on the left) and its poset of derivatives (on the right). These automata are infinite-state, but we write them symbolically, with atoms $a, b, \ldots \in \mathbb{A}$. The states are anonymous, but we write the current value of a register inside the state. The posets may not be orbit-finite, in which case we depict a small, indicative part. Orbits are depicted as grey rectangles.

Deterministic: $\mathcal{L}_\mathrm{d} \coloneqq \{awa \mid a \in \mathbb{A}, w \in \Sigma^*\}$ (First symbol equals last symbol)

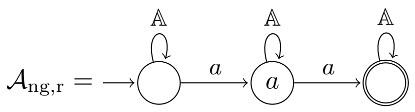


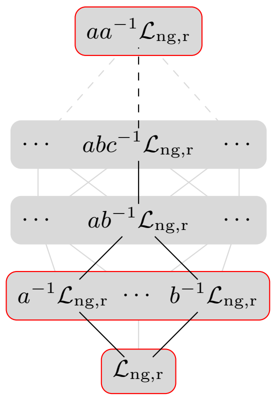


$\mathcal{L}_\mathrm{d}$
$aa^{-1}\mathcal{L}_\mathrm{d}$
$bb^{-1}\mathcal{L}_\mathrm{d}$
$\cdots$
$a^{-1}\mathcal{L}_\mathrm{d}$
$b^{-1}\mathcal{L}_\mathrm{d}$
$\cdots$

Note that the derivatives $a^{-1}\mathcal{L}_\mathrm{d}, b^{-1}\mathcal{L}_\mathrm{d}, \ldots$ are in the same orbit. In total there are three orbits of derivatives and they correspond to the three 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: $\mathcal{L}_\mathrm{ng,r} \coloneqq \{uavaw \mid u, v, w \in \mathbb{A}^*, a \in \mathbb{A}\}$ (Some atom occurs twice)

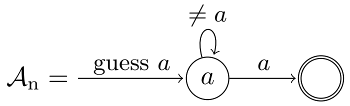


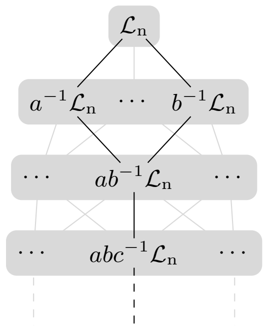


$aa^{-1}\mathcal{L}_\mathrm{ng,r}$
$\cdots$ $abc^{-1}\mathcal{L}_\mathrm{ng,r}$ $\cdots$
$\cdots$ $ab^{-1}\mathcal{L}_\mathrm{ng,r}$ $\cdots$
$a^{-1}\mathcal{L}_\mathrm{ng,r}$ $\cdots$ $b^{-1}\mathcal{L}_\mathrm{ng,r}$
$\mathcal{L}_\mathrm{ng,r}$

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, the derivative $ab^{-1}\mathcal{L}_\mathrm{ng,r}$ 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. This will be the main technique of constructing residual automata in Theorem 2.

Nondeterministic: $\mathcal{L}_\mathrm{n} \coloneqq \{wa \mid a \text{ not in } w\}$ (Last letter is unique)

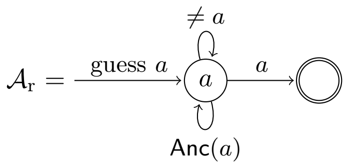


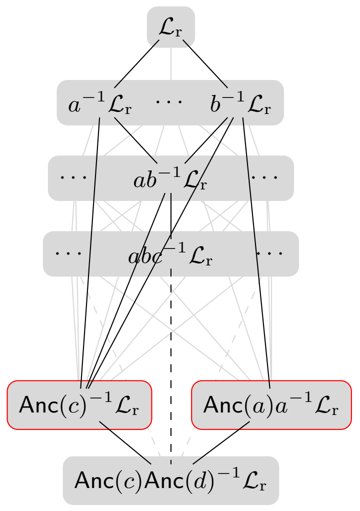


$\mathcal{L}_\mathrm{n}$
$a^{-1}\mathcal{L}_\mathrm{n}$ $\cdots$ $b^{-1}\mathcal{L}_\mathrm{n}$
$\cdots$ $ab^{-1}\mathcal{L}_\mathrm{n}$ $\cdots$
$\cdots$ $abc^{-1}\mathcal{L}_\mathrm{n}$ $\cdots$

The poset $\mathsf{Der}(\mathcal{L})$ has an infinite descending chain of languages (with an increasing support). The derivative $a^{-1}\mathcal{L}_\mathrm{n}$ is again a union of smaller languages: $a^{-1}\mathcal{L}_\mathrm{n} = \bigcup_{b \neq a} ab^{-1}\mathcal{L}_\mathrm{n}$. (We have omitted languages like $aa^{-1}\mathcal{L}_\mathrm{n}$, they only differ from $a^{-1}\mathcal{L}_\mathrm{n}$ on the empty word.) The language cannot be accepted by a residual automaton, which will follow from Theorem 2.

Residual: (Last letter is unique with anchor)

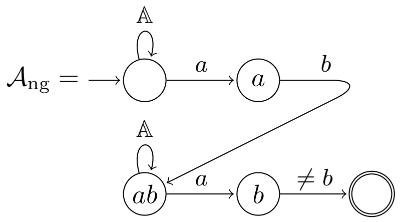


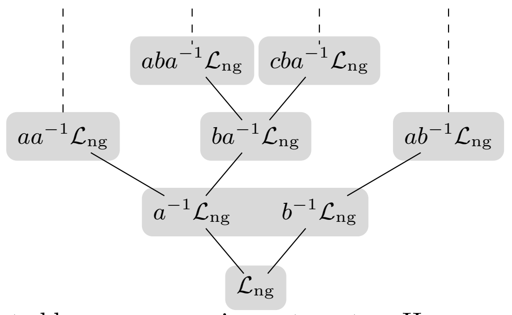


$\mathcal{L}_\mathrm{r}$
$a^{-1}\mathcal{L}_\mathrm{r}$
$\cdots$
$b^{-1}\mathcal{L}_\mathrm{r}$
$\cdots$
$ab^{-1}\mathcal{L}_\mathrm{r}$
$\cdots$
$\cdots$
$abc^{-1}\mathcal{L}_\mathrm{r}$
$\cdots$
$\mathsf{Anc}(c)^{-1}\mathcal{L}_\mathrm{r}$
$\mathsf{Anc}(a)a^{-1}\mathcal{L}_\mathrm{r}$
$\mathsf{Anc}(c)\mathsf{Anc}(d)^{-1}\mathcal{L}_\mathrm{r}$

Consider the alphabet $\Sigma = \mathbb{A} \cup \{\mathsf{Anc}(a) \mid a \in \mathbb{A}\}$ and 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 be obtained by 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 $\mathsf{Anc}(a)a^{-1}\mathcal{L}_\mathrm{r} = \{\epsilon\}$.

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





$aba^{-1}\mathcal{L}_\mathrm{ng}$
$cba^{-1}\mathcal{L}_\mathrm{ng}$
$aa^{-1}\mathcal{L}_\mathrm{ng}$
$ba^{-1}\mathcal{L}_\mathrm{ng}$
$ab^{-1}\mathcal{L}_\mathrm{ng}$
$a^{-1}\mathcal{L}_\mathrm{ng}$
$b^{-1}\mathcal{L}_\mathrm{ng}$
$\mathcal{L}_\mathrm{ng}$

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 return to this example after Theorem 2.

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 2). 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 $\mathcal{P}(Z)$ for an arbitrary nominal set Z. This is a Boolean algebra of which the operations $\wedge, \vee, \neg$

are all equivariant maps [18]. Moreover, the finitely supported union

$$\bigcup \colon \mathcal{P}(\mathcal{P}(Z)) \to \mathcal{P}(Z),$$

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 denote this union by $\bigvee$, set inclusion by $\leq$, and strict inclusion by $<$.

Definition 4. *Given a nominal set Z and $X \subseteq \mathcal{P}(Z)$ equivariant*[6]*, we define the set generated by X as*

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

Remark 1. The set $\langle X \rangle$ is closed under the operation $\bigvee$, and moreover is the smallest equivariant set closed under $\bigvee$ containing X.

Definition 5. *Let $X \subseteq \mathcal{P}(Z)$ equivariant and $x \in X$, we say that x is* join-irreducible *in X if it is non-empty and*

$$x = \bigvee \mathfrak{x} \quad \implies \quad \exists x_0 \in \mathfrak{x} \text{ such that } x = x_0, \tag{1}$$

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 2. Condition (1) can be equivalently stated as

$$\forall x_0 \in \mathfrak{x} \,.\, x_0 < x \quad \implies \quad \bigvee \mathfrak{x} < x. \tag{1'}$$

Remark 3. In the theory of lattices and order, join-irreducible elements are usually defined only for a lattice (see, for example, [13]). 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 $\mathcal{P}(\Sigma^*)$.

Remark 4. The notion of join-irreducible, as we have defined here, corresponds to the notion of *prime* in [8,15,29]. 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 results roughly follow [13, Lemma 2.45].

[6] 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.

Lemma 1. *Let $X \subseteq \mathcal{P}(Z)$ be an orbit-finite and equivariant set.*

1. *Let $a \in X, b \in \mathcal{P}(Z)$ 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 $a = \bigvee \{x \in X \mid x$ join-irreducible in X 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 $b = \bigvee T \in \mathcal{P}(Z)$. 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. □

Corollary 1. *Let $X \subseteq \mathcal{P}(Z)$ 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 2. *Let $Y \subseteq X \subseteq \mathcal{P}(Z)$ 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 $x_0 \in \mathfrak{x}$. □

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

Corollary 2. *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 2. *Given a language $\mathcal{L} \in \mathcal{P}(\Sigma^*)$, the following are equivalent:*

1. *$\mathcal{L}$ is accepted by a residual automaton.*
2. *There is some orbit-finite set $J \subseteq \mathsf{Der}(\mathcal{L})$ 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 state: $J = \{\mathcal{L}(q) \mid q \in \mathcal{A}\}$. 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 2 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 construct the following automaton:

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

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. Second, we will show that each state $q = w^{-1}\mathcal{L}$ accepts its language: $\mathcal{L}(q) = w^{-1}\mathcal{L}$. This is shown with induction as follows.

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

At step (i) we have used the induction hypothesis (u is a smaller 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. To finish the proof, note that $\mathcal{L} = \bigvee \left\{w^{-1}\mathcal{L} \mid w^{-1}\mathcal{L} \leq \mathcal{L}\right\}$, since the join-irreducible languages generate all languages. In particular, the initial states (together) accept $\mathcal{L}$. □

Corollary 3. *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 2, 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. □

There is a crucial difference to the case of finite alphabets. 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 2) 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 2.

Remark 5. We note that the canonical automaton enjoys the so-called *strongly-consistency* property [15]. This means that *every* characterising word for a state q leads to the state q. Formally, if $\mathcal{L}(q) = w^{-1}\mathcal{L}(\mathcal{A})$, then $q \in \delta(I, w)$.

Example 1. In this example we show that residual automata can also be used to compress deterministic automata. The language

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

(a zero amount of bs is also accepted) can be accepted by a deterministic automaton of 4 orbits, and this is minimal. The minimal residual automaton, however, has only 2 orbits, given by the join-irreducible languages:

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

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 2. 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

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

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 the set $\mathsf{Der}(\mathcal{L}_{\mathrm{n}})$ has *no join-irreducible elements.* In this case $\mathsf{JI}(\mathsf{Der}(\mathcal{L}_{\mathrm{n}}))$ does not generate $\mathsf{Der}(\mathcal{L}_{\mathrm{n}})$ and so by Theorem 2 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 a infinite ascending chains. 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. So the set $\mathsf{JI}(\mathsf{Der}(\mathcal{L}_{\mathrm{ng}}))$ is *not orbit-finite*. By Theorem 2, we conclude that there is no residual automaton accepting $\mathcal{L}_{\mathrm{ng}}$.

Remark 6. For arbitrary (nondeterministic) languages there is also a characterisation. Namely, $\mathcal{L}$ is accepted by an automaton iff there is an orbit-finite set $Y \subseteq \mathcal{P}(\Sigma^*)$ 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. To adapt to the situation, 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 6. *Let $X \subseteq \mathcal{P}(Z)$ equivariant and $x \in X$, we say that x is* join-irreducible^{-} in X *if*

$$x = \bigvee \mathfrak{x} \quad\implies\quad \exists x_0 \in \mathfrak{x} \text{ such that } x = x_0,$$

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

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

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 3. *Given a language $\mathcal{L} \in \mathcal{P}(\Sigma^*)$, 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 2. However, we need a slightly different definition of the canonical automaton. It is defined as follows.

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

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.

Lemma 3. *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.*

From this fact we can derive the following folklore result:

Corollary 4. *The transition relation δ of a non-guessing automaton can be equivalently be described as a function*

$$\delta \colon Q \times \Sigma \to \mathcal{P}_{\text{fin}}(Q),$$

where $\mathcal{P}_{\text{fin}}(Q)$ is the set of all 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 3, instead of uniformly supported unions.

5 (Un)Decidability Results

In this section we prove two results. 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 [5]. Second, a negative result: deciding whether a automaton is residual is undecidable.

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 [24,25].

Proposition 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 (inductively) 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 2. *The problem of determining whether nominal automata are residual is undecidable.*

Proof. The construction is inspired by [15, Proposition 8.4].[7] 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:

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

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

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

See Figure 2 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.

[7] They prove that checking residuality for NFAs is PSPACE-complete via a reduction from universality. However, instead of using NFAs, they use a union of n DFAs. This would not work in the nominal setting.

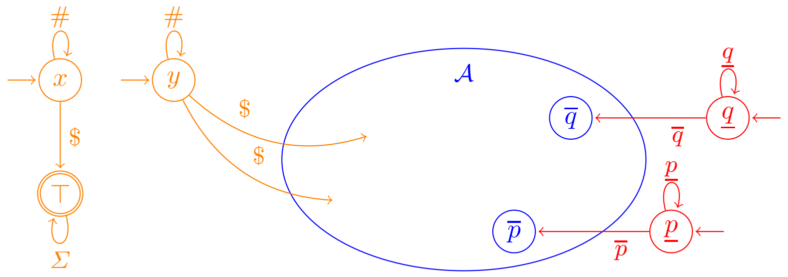


Fig. 2. Sketch of the automaton $\mathcal{A}'$ constructed in the proof of Proposition 2.

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 2). 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 $a \in \Sigma$, $\overline{q}$, $\underline{q}$ 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. □

We emphasise that the above result also holds for non-guessing automata, as the construction does not introduce any guessing and universality for non-guessing nondeterministic nominal automata is undecidable.

Remark 7. 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 1). We conjecture that equivalence remains undecidable for residual automata.

6 Exact learning

In our previous paper on learning nominal automata [29], we were able to learn residual automata accepting deterministic languages. Moreover, we observed by experimentations that the algorithm was also able to learn non-deterministic residual languages in specific cases. However, several questions on nominal languages remained open, most importantly:

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

In this section we show that the theory developed in previous sections will allow us to answer these questions. Moreover, we will discuss how two apparently distant learning problems – concerning learning nominal and probabilistic languages – are actually tightly related.

6.1 Finite characterisation of languages

Hankel matrices are important tools used in learning theory to characterise languages (we borrow the terminology from the weighted setting [3]). Given a language $\mathcal{L}$, its Hankel matrix $\mathbb{H}_{\mathcal{L}}$ is an infinite matrix in $2^{\Sigma^* \times \Sigma^*}$ such that $\mathbb{H}_{\mathcal{L}}(u, v) = \mathcal{L}(uv)$.[8] The crucial property of $\mathbb{H}_{\mathcal{L}}$ is that a row $\mathbb{H}_{\mathcal{L}}(u)$ is precisely a derivative $u^{-1}\mathcal{L}$, and two words u and u' are in the same Myhill-Nerode equivalence class if and only if $\mathbb{H}_{\mathcal{L}}(u) = \mathbb{H}_{\mathcal{L}}(u')$.

Intuitively, $\mathbb{H}_{\mathcal{L}}$ consists of all possible observations that can be made about a language. An important question in learning theory is under which conditions $\mathcal{L}$ can be characterised via a *finite* sub-matrix of $\mathbb{H}_{\mathcal{L}}$. This, e.g., is crucial for Angluin-style automata learning algorithms, which attempt at building an automaton from a finite set of observations.

In the regular languages setting this is *always* possible. In fact, regular languages have finitely-many derivatives, so $\mathbb{H}_{\mathcal{L}}$ has finitely-many distinct rows and we only need a finite amount of words (columns of $\mathbb{H}_{\mathcal{L}}$) to distinguish them. In the nominal setting this question has a more nuanced answer.

We now give sufficient and necessary conditions for when the rows of $\mathbb{H}_{\mathcal{L}}$ form an orbit-finite – thus finitely representable – set, or can be generated by such a set. We restate Theorem 1 in terms of the Hankel matrix:

Corollary 5. *$\mathbb{H}_{\mathcal{L}}$ has orbit-finitely-many rows iff $\mathcal{L}$ is deterministic.*

In general, $\mathbb{H}_{\mathcal{L}}$ has orbit-infinitely-many distinct rows. In the residual case, however, we still have finite representability, by Theorem 2.

Corollary 6. *The rows of $\mathbb{H}_{\mathcal{L}}$ are generated by an orbit-finite subset if and only if $\mathcal{L}$ is residual.*

Using Theorem 3 (and Lemma 3), we obtain the non-guessing case.

Corollary 7. *If $\mathcal{L}$ is non-guessing, then the rows of $\mathbb{H}_{\mathcal{L}}$ are generated by* finite *unions of an orbit-finite subset.*

These finiteness results unfortunately do not tell us *how* to actually obtain a finite sub-matrix. This is always possible in the deterministic case. In the general residual case, however, no efficient algorithm is known. In fact, in the following section we will show that an Angluin-style learning algorithm *does not* work.

[8] Here we use the $\mathcal{L} \colon \Sigma^* \to 2$ as opposed to $\mathcal{L} \in \mathcal{P}(\Sigma^*)$. The two perspectives are equivalent.

6.2 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 $\mathcal{L}$. 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 [23] or [36] for details).

At any stage of the learning algorithm, the learner will store an *observation table* $\mathcal{T}$, that is a finite sub-matrix of $\mathbb{H}_{\mathcal{L}}$ filled in via membership queries. The table $\mathcal{T}$ has a very specific shape: rows are labelled by a prefix-closed set $U \subseteq \Sigma^*$. As mentioned in the previous section, we have that $\mathcal{T}(u)$ is an approximation of $u^{-1}\mathcal{L}$. 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 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 $u \in U, a \in \Sigma$. If the table is closed,[9] $\mathsf{L}^\star$ is able to construct an automaton. 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. When the table is not closed, then a derivative is missing, and a corresponding row needs to be added.

The nominal setting. In [29] we have given nominal versions of $\mathsf{L}^\star$ and $\mathsf{NL}^\star$. They seamlessly extend the original algorithms by operating on orbit-finite sets. The key result is that one can learn a (infinite-state) nominal automaton with finitely many queries:

Theorem 4 ([29]). *If $\mathcal{L}$ is deterministic, then nominal $\mathsf{L}^\star$ and nominal $\mathsf{NL}^\star$ converge, and output the canonical deterministic and residual automaton accepting $\mathcal{L}$, respectively.*

This result can be explained by Corollary 5. Nominal $\mathsf{L}^\star$ is even quite efficient: only $\mathcal{O}(nk)$ equivalence queries are needed and one only needs $\mathcal{O}(n)$ rows in the

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

table (ignoring the length of counterexamples), where n is the number of orbits of the canonical automaton accepting $\mathcal{L}$ and k the number of registers [28].

Now, while there is no hope of convergence in the general nondeterministic case, as the information in the Hankel matrix cannot be finitely represented, Corollary 6 appears to give us some hope for the residual case. However, we will show via an example that residuality *does not* imply convergence.

Example 3. The issue can be shown with an example where the teacher is adversarial (but is still playing according to the rules).

Consider the language $\mathcal{L}_{\mathrm{r}}$ from Section 3. Suppose we want to learn it with the nominal versions of $\mathsf{L}^\star$ or $\mathsf{NL}^\star$. The algorithm starts querying and observes the derivatives $\epsilon^{-1}\mathcal{L}(= \mathcal{L})$ and the (orbit of the) one-letter extension $a^{-1}\mathcal{L}$. The table is not closed (in either algorithm) since $a^{-1}\mathcal{L} < \mathcal{L}$, and so it adds the one-letter extension. Then, it detects the derivative $ab^{-1}\mathcal{L}$, which still does not close the table, after which it finds $abc^{-1}\mathcal{L}$. Since each derivative is strictly smaller then the previous one, the table will never be closed, unless the learner queries words starting with $\mathsf{Anc}(a)$. This will of course happen quite quickly, but we could change the example by replacing $\mathsf{Anc}(a)$ by a long, hard-to-find, word. In such a case, the teacher will not have to reveal such a word as long as the learner is still incorrect on the languages $\mathcal{L} > a^{-1}\mathcal{L} > ab^{-1}\mathcal{L} > \cdots$. This shows that neither $\mathsf{L}^\star$ or $\mathsf{NL}^\star$ will terminate.

The only way to learn $\mathcal{L}_{\mathrm{r}}$ seems to be to exhaustively enumerate longer and longer words, instead of adding only the missing rows. Although this will provide a learning algorithm (by Corollary 6), it is a highly inefficient one. Even worse, characterising words may be of exponential length (see [15, Proposition 6.3]).

We can see that this problem is already present in the learning algorithm $\mathsf{NL}^\star$ for RFSA. In fact, the complexity of the $\mathsf{NL}^\star$ algorithm is given in terms of the minimal DFA, which may be exponential in the size of the canonical residual one. Similarly, in the $\mathsf{AL}^\star$ algorithm (for alternating automata [2]), the complexity is given in terms of the minimal DFA.

Ideally, an $\mathsf{L}^\star$-type algorithm for residual automata should not depend on the size (or even existence) of the deterministic automaton. Instead, we would like an algorithm of which the complexity depends only on the length of characterising words. To the best of our knowledge, no such algorithm exists.

6.3 Relation to Probabilistic Automata

Residual nominal automata and residual probabilistic automata enjoy similar properties. In fact, they define a class of languages strictly in between the deterministic and nondeterministic ones. Although the residuality property is defined on automata, both classes of languages admit a machine-independent characterisation. In the case of nominal languages, this is in terms on join-irreducible languages (Theorem 2), and in the case of probabilistic automata this is in terms of extremal languages which generate other languages by *convex combinations*.

In learning of probabilistic automata, we can learn the deterministic ones [10,11,14]. But learning residual ones or general ones is still open. For the general case, some negative results exist [35], which suggests that learning is not possible in an efficient way. This was the reason of Denis et. al. to consider residual automata in the first place. The machine-independent characterisation seems to be useful for learning, as the necessary information can be found in $\mathsf{Der}(\mathcal{L})$ (or the Hankel matrix). Unfortunately, it is not yet clear how to obtain the right finite sub-matrix of the Hankel matrix. But, here too, once the right finite part is obtained, a canonical construction will give the correct residual automaton.

7 Conclusions

In this paper we have investigated a subclass of nondeterministic automata over infinite alphabets. This class is relevant in the context of data languages, but also in the context of learning and residuality in general. Remarkably, the class admits canonical automata, where states are defined by join-irreducible elements. Moreover, universality becomes decidable, in contrast to undecidability in the general nondeterministic case. In the context of learning, we show that residual languages can be characterised via a finite number of observations. Unfortunately, we also show that this does not imply convergence of Angluin-style learning.

Future and related work. Another aspect of interest for us is closure properties of nominal residual languages. This class is not closed under complement (the complement of $\mathcal{L}_{\mathrm{ng,r}}$ is not even accepted by a nondeterministic automaton) or intersection ($\mathcal{L}_{\mathrm{r}} \cap \mathbb{A}^* = \mathcal{L}_{\mathrm{n}}$, a nondeterministic language). However, we conjecture that it is closed under concatenation and Kleene star. To deal with intersections, one could investigate properties of *alternating* residual nominal automata. For finite alphabets, these are discussed in [2] in the context of learning. We note, however, that such automata are unlikely to have canonical models, as the set of generators cannot be chosen uniformly (i.e., there is no analogue of $\mathsf{JI}(X)$ when considering both unions and intersections).

We also 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 [9,12,30,34].

Other related work are nominal languages/expressions with an explicit notion of binding [16,26,27,34]. Although these are sub-classes of nominal languages, binding is an important construct, e.g., to represent resource-allocation. Availability of a notion of derivates [26] suggests that residuality may prove beneficial for learning these languages.

Residual automata over finite alphabets also have a categorical characterisation [31]. We see no obstructions in generalising those results to nominal sets. This would amount to finding the right notion of nominal join-semilattice, with either finitely or uniformly supported joins.

Finally, in [17,18] 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.

Acknowledgements. We would like to thank Gerco van Heerdt for providing examples similar to that of $\mathcal{L}_{\mathrm{r}}$ in the context of probabilistic automata. We thank Borja Balle for many references on residual probabilistic languages. Lastly, we thank Henning Urbat for discussions on nominal lattice theory.

References

1. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. **75**(2), 87–106 (1987)
2. Angluin, D., Eisenstat, S., Fisman, D.: Learning regular languages via alternating automata. In: IJCAI. pp. 3308–3314. AAAI Press (2015)
3. Balle, B., Mohri, M.: Learning weighted automata. In: CAI. pp. 1–21 (2015). https://doi.org/10.1007/978-3-319-23021-4_1
4. Berndt, S., Liśkiewicz, M., Lutter, M., Reischuk, R.: Learning residual alternating automata. In: AAAI. pp. 1749–1755 (2017), `http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14748`
5. Bojańczyk, M.: Slightly Infinite Sets. Draft September 6, 2019 (2019), `https://www.mimuw.edu.pl/~bojan/paper/atom-book`
6. Bojańczyk, M., Klin, B., Lasota, S.: Automata theory in nominal sets. Logical Methods in Computer Science **10**(3) (2014)
7. Bojańczyk, M., Toruńczyk, S.: On computability and tractability for infinite sets. In: LICS. pp. 145–154. ACM (2018)
8. Bollig, B., Habermehl, P., Kern, C., Leucker, M.: Angluin-style learning of NFA. In: IJCAI. pp. 1004–1009 (2009)
9. Bollig, B., Habermehl, P., Leucker, M., Monmege, B.: A robust class of data languages and an application to learning. Logical Methods in Computer Science **10**(4) (2014). https://doi.org/10.2168/LMCS-10(4:19)2014
10. Castro, J., Gavalda, R.: Learning probability distributions generated by finite-state machines. In: Topics in Grammatical Inference, pp. 113–142. Springer (2016)
11. Clark, A., Thollard, F.: Pac-learnability of probabilistic deterministic finite state automata. J. Mach. Learn. Res. **5**, 473–497 (2004)
12. Colcombet, T.: Unambiguity in automata theory. In: Descriptional Complexity of Formal Systems - 17th International Workshop, DCFS 2015, Waterloo, ON, Canada, June 25-27, 2015. Proceedings. pp. 3–18 (2015). https://doi.org/10.1007/978-3-319-19225-3_1
13. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order, Second Edition. Cambridge University Press (2002). https://doi.org/10.1017/CBO9780511809088, `https://doi.org/10.1017/CBO9780511809088`
14. Denis, F., Esposito, Y.: On rational stochastic languages. Fundam. Inform. **86**(1-2), 41–77 (2008)
15. Denis, F., Lemay, A., Terlutte, A.: Residual finite state automata. Fundam. Inform. **51**(4), 339–368 (2002)
16. Gabbay, M.J., Ciancia, V.: Freshness and name-restriction in sets of traces with names. In: FOSSACSs. pp. 365–380 (2011). https://doi.org/10.1007/978-3-642-19805-2_25
17. Gabbay, M.J., Gabbay, M.: 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). https://doi.org/10.1016/j.apal.2016.10.001

18. Gabbay, M.J., Litak, T., Petrisan, D.: Stone duality for nominal boolean algebras with N. In: CALCO. Lecture Notes in Computer Science, vol. 6859, pp. 192–207. Springer (2011)
19. Grigore, R., Distefano, D., Petersen, R.L., Tzevelekos, N.: Runtime verification based on register automata. In: TACAS. Lecture Notes in Computer Science, vol. 7795, pp. 260–276. Springer (2013)
20. Howar, F., Jonsson, B., Vaandrager, F.W.: Combining black-box and white-box techniques for learning register automata. In: Computing and Software Science, Lecture Notes in Computer Science, vol. 10000, pp. 563–588. Springer (2019)
21. Kaminski, M., Francez, N.: Finite-memory automata. Theor. Comput. Sci. **134**(2), 329–363 (1994)
22. Kaminski, M., Zeitlin, D.: Finite-memory automata with non-deterministic reassignment. Int. J. Found. Comput. Sci. **21**(5), 741–760 (2010). https://doi.org/10.1142/S0129054110007532
23. Kearns, M.J., Vazirani, U.V.: An Introduction to Computational Learning Theory. MIT Press (1994), `https://mitpress.mit.edu/books/introduction-computational-learning-theory`
24. Klin, B., Szynwelski, M.: SMT solving for functional programming over infinite structures. In: MSFP. EPTCS, vol. 207, pp. 57–75 (2016)
25. Kopczynski, E., Toru'nczyk, S.: LOIS: syntax and semantics. In: POPL. pp. 586–598. ACM (2017)
26. Kozen, D., Mamouras, K., Petrisan, D., Silva, A.: Nominal kleene coalgebra. In: ICAL. pp. 286–298 (2015). https://doi.org/10.1007/978-3-662-47666-6_23
27. Kurz, A., Suzuki, T., Tuosto, E.: On nominal regular languages with binders. In: FOSSACS. pp. 255–269 (2012). https://doi.org/10.1007/978-3-642-28729-9_17
28. Moerman, J.: Nominal Techniques and Black Box Testing for Automata Learning. Ph.D. thesis, Radboud University, Nijmegen, The Netherlands (2019), `http://hdl.handle.net/2066/204194`
29. Moerman, J., Sammartino, M., Silva, A., Klin, B., Szynwelski, M.: Learning nominal automata. In: POPL. pp. 613–625. ACM (2017)
30. Mottet, A., Quaas, K.: The containment problem for unambiguous register automata. In: STACS. pp. 53:1–53:15 (2019). https://doi.org/10.4230/LIPIcs.STACS.2019.53
31. Myers, R.S.R., Adámek, J., Milius, S., Urbat, H.: Coalgebraic constructions of canonical nondeterministic automata. Theor. Comput. Sci. **604**, 81–101 (2015)
32. Neven, F., Schwentick, T., Vianu, V.: Finite state machines for strings over infinite alphabets. ACM Trans. Comput. Log. **5**(3), 403–435 (2004)
33. Pitts, A.M.: Nominal sets: Names and symmetry in computer science. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press (2013)
34. Schröder, L., Kozen, D., Milius, S., Wißmann, T.: Nominal automata with name binding. In: FOSSACS. pp. 124–142 (2017). https://doi.org/10.1007/978-3-662-54458-7_8
35. Terwijn, S.: On the learnability of hidden markov models. In: ICGI. Lecture Notes in Computer Science, vol. 2484, pp. 261–268. Springer (2002)
36. Vaandrager, F.W.: Model learning. Commun. ACM **60**(2), 86–95 (2017)