Positive Data Languages

Florian Frank
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Stefan Milius
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Henning Urbat
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Abstract

Positive data languages are languages over an infinite alphabet closed under possibly non-injective renamings of data values. Informally, they model properties of data words expressible by assertions about equality, but not inequality, of data values occurring in the word. We investigate the class of positive data languages recognizable by nondeterministic orbit-finite nominal automata, an abstract form of register automata introduced by Bojańczyk, Klin, and Lasota. As our main contribution we provide a number of equivalent characterizations of that class in terms of positive register automata, monadic second-order logic with positive equality tests, and finitely presentable nondeterministic automata in the categories of nominal renaming sets and of presheaves over finite sets.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Data Languages, Register Automata, MSO, Nominal Sets, Presheaves

Digital Object Identifier 10.4230/LIPIcs.MFCS.2023.48

Related Version *Full Version with omitted details and proofs*: `https://arxiv.org/abs/2304.12947`

Funding *Florian Frank*: Supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as part of the Research and Training Group 2475 "Cybercrime and Forensic Computing" (grant number 393541319/GRK2475/1-2019).
Stefan Milius: Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 419850228.
Henning Urbat: Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 470467389.

Acknowledgements The authors wish to thank Bartek Klin for pointing out the example in Remark 2.10.

1 Introduction

Automata over infinite alphabets provide a simple computational model for reasoning about structures involving *data* such as nonces [21], URLs [3], or values in XML documents [25]. Consider, for instance, the (infinite) set $\mathbb{A}$ of admissible user IDs for a server. The sequence of all user logins within a given time period then forms a finite word $a_1 \cdots a_n \in \mathbb{A}^\star$ over the infinite alphabet $\mathbb{A}$, and behaviour patterns may be modelled as data languages over $\mathbb{A}$, e.g.

$$L_0 = \{\, a_1 \cdots a_n \in \mathbb{A}^\star \mid a_i \neq a_n \text{ for all } i < n \,\} \qquad (\text{“last user has not logged in before”}),$$
$$L_1 = \{\, a_1 \cdots a_n \in \mathbb{A}^\star \mid a_i = a_j \text{ for some } i \neq j \,\} \qquad (\text{“some user has logged in twice”}).$$

Both L_0 and L_1 involve assertions about equality, or inequality, of data values (here, user IDs). However, asserting *inequality* is sometimes considered problematic and thus undesired. For example, since users may have multiple IDs, a logfile $a_1 \dots a_n \in L_0$ does not actually guarantee that the last user has not logged in before. In contrast, if $a_1 \dots a_n \in L_1$, then it is guaranteed that some user has indeed logged in twice. The structural difference between the

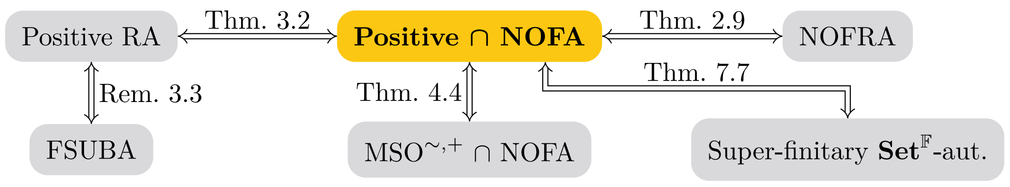

Figure 1 Equivalent characterizations of positive NOFA-recognizable languages.

two languages is that L_1 is closed under arbitrary renamings $\rho\colon \mathbb{A} \to \mathbb{A}$ (i.e. $a_1 \cdots a_n \in L_1$ implies $\rho(a_1)\cdots\rho(a_n) \in L_1$), taking into account possible identification of data values, while L_0 is only closed under injective (equivalently bijective) renamings. We refer to languages with the former, stronger closure property as *positive data languages*. Intuitively, such languages model properties of data words expressible by positive statements about equality of data values. It is one of the goals of our paper to turn this into a theorem.

For that purpose, we build on the abstract account of data languages and their automata based on the theory of nominal sets [13, 27], initiated by the work of Bojańczyk, Klin, and Lasota [6]. Specifically, we investigate *nondeterministic orbit-finite nominal automata* (*NOFA*), the nominal version of classical nondeterministic finite automata. We approach the class of *positive* NOFA-recognizable data languages from several different perspectives, ranging from concrete to more abstract and conceptual, and establish the equivalent characterizations summarized in Figure 1. In more detail, our main contributions are as follows.

Register automata. NOFAs are known to be expressively equivalent to register automata [17, 19], i.e. finite automata that can memorize data values using a fixed number of registers and test the input for (in)equality with previously stored values. Restricting transitions to positive equality tests leads to *positive register automata*, which correspond to *finite-state unification-based automata* (*FSUBA*) [18, 32] and are shown to capture precisely positive NOFA-recognizable languages (Theorem 3.2 and Remark 3.3). On the way, we isolate a remarkable property of this language class: while NOFAs generally require the ability to guess data values during the computation to reach their full expressive strength, guessing and non-guessing NOFA are equivalent for positive data languages (Theorem 2.17).

Monadic second-order logic. As illustrated above, positive data languages model (only) positive assertions about the equality of data values. To substantiate this intuition, we employ monadic second-order logic ($\mathrm{MSO}^{\sim}$) over data words [4, 9, 25], an extension of classical MSO with equality tests for data values, and consider its restriction $\mathrm{MSO}^{\sim,+}$ to positive equality tests. While this logic is more expressive than NOFA, we show that within the class of NOFA-recognizable languages it models exactly the positive languages (Theorem 4.4).

Categorical perspective. The classical notion of nondeterministic finite automata can be categorified by replacing the finite set of states with a finitely presentable object of a category $\mathscr{C}$. For example, NOFAs are precisely nondeterministic $\mathscr{C}$-automata for $\mathscr{C}$ = nominal sets. Apart from the latter category, several other toposes have been proposed as abstract foundations for reasoning about names (data values), most prominently the category of *nominal renaming sets* [12], the category $\mathbf{Set}^{\mathbb{I}}$ of presheaves over finite sets and injective maps [31], and the category $\mathbf{Set}^{\mathbb{F}}$ of presheaves over finite sets and all maps (equivalently, finitary set functors) [10]. It is thus natural to study nondeterministic automata in the latter three categories, viz. *nondeterministic orbit-finite renaming automata* (*NOFRA*), *nondeterministic super-finitary* $\mathbf{Set}^{\mathbb{I}}$*-automata* and *nondeterministic super-finitary* $\mathbf{Set}^{\mathbb{F}}$*-automata*. Our final contribution is a classification of their expressive power: we show that $\mathbf{Set}^{\mathbb{I}}$-automata are equivalent to NOFAs, while both NOFRAs and $\mathbf{Set}^{\mathbb{F}}$-automata capture positive NOFA-recognizable languages (Theorems 2.9 and 7.7). Hence, both nominal and presheaf-based automata are able to recognize positive and all NOFA-recognizable languages, respectively.

2 Nominal Automata and Positive Data Languages

For the remainder of the article, we fix a countably infinite set $\mathbb{A}$ of *data values*, a.k.a. *names* or *atoms*. The goal is to study positive data languages, that is, languages of finite words over $\mathbb{A}$ closed under arbitrary renamings. This is achieved via the framework of nominal (renaming) sets [12, 13, 27].

2.1 Nominal Sets and Nominal Renaming Sets

A *renaming* is a finite map $\rho\colon \mathbb{A} \to \mathbb{A}$, that is, $\rho(a) = a$ for all but finitely many $a \in \mathbb{A}$. We let $\mathsf{Fin}(\mathbb{A})$ denote the monoid of renamings, with multiplication given by composition, and $\mathsf{Perm}(\mathbb{A})$ its subgroup given by *finite permutations*, i.e. bijective renamings. For $M \in \{\,\mathsf{Perm}(\mathbb{A}), \mathsf{Fin}(\mathbb{A})\,\}$ an *M-set* is a set X equipped with a monoid action $M \times X \to X$, denoted $(\rho, x) \mapsto \rho \cdot x$. A subset $S \subseteq \mathbb{A}$ is a *support* of $x \in X$ if for every $\rho, \sigma \in M$ such that $\rho|_S = \sigma|_S$ one has $\rho \cdot x = \sigma \cdot x$. Informally, consider X as a set of syntactic objects (e.g. words, trees, λ-terms) whose description may involve free names from S. A *nominal M-set* is an M-set where every element x has a finite support. This implies that x has a least finite support $\mathsf{supp}\, x \subseteq \mathbb{A}$. A name $a \in \mathbb{A}$ is *fresh* for x, denoted $a \mathrel{\#} x$, if $a \notin \mathsf{supp}\, x$.

Nominal $\mathsf{Perm}(\mathbb{A})$-sets are called *nominal sets*, and nominal $\mathsf{Fin}(\mathbb{A})$-sets are called *nominal renaming sets*. A nominal renaming set X can be regarded as a nominal set by restricting its $\mathsf{Fin}(\mathbb{A})$-action to a $\mathsf{Perm}(\mathbb{A})$-action. The least supports of an element $x \in X$ w.r.t. both actions coincide [11, Thm. 4.8], so the notation $\mathsf{supp}\, x$ is unambiguous.

A subset X of a nominal M-set Y is *M-equivariant* if $\rho \cdot x \in X$ for all $x \in X$ and $\rho \in M$. More generally, a map $f\colon X \to Y$ between nominal M-sets is *M-equivariant* if $f(\rho \cdot x) = \rho \cdot f(x)$ for all $x \in X$ and $\rho \in M$. This implies $\mathsf{supp}\, f(x) \subseteq \mathsf{supp}\, x$ for all $x \in X$.

We write $X \times Y$ for the cartesian product of nominal M-sets with componentwise action, and $\coprod_{i\in I} X_i$ for the coproduct (disjoint union) with action inherited from the summands.

Given a nominal set X, the *orbit* of an element $x \in X$ is the set $\{\pi \cdot x : \pi \in \mathsf{Perm}(\mathbb{A})\}$. The orbits form a partition of X. A nominal set is *orbit-finite* if it has only finitely many orbits. A nominal renaming set is *orbit-finite* if it is orbit-finite as a nominal set.

▶ **Example 2.1.** The set $\mathbb{A}$ with the $\mathsf{Fin}(\mathbb{A})$-action $\rho \cdot a = \rho(a)$ is a nominal renaming set, as is the set $\mathbb{A}^\star$ of finite words over $\mathbb{A}$ with $\rho \cdot w = \rho^\star(w) = \rho(a_1) \cdots \rho(a_n)$ for $w = a_1 \cdots a_n$. The least support of $a_1 \cdots a_n \in \mathbb{A}^\star$ is the set $\{a_1, \dots, a_n\}$. The set $\mathbb{A}^\star$ has infinitely many orbits; its equivariant subsets $\mathbb{A}^n$ (words of a fixed length n) are orbit-finite. For instance, $\mathbb{A}^2$ has the two orbits $\{aa : a \in \mathbb{A}\}$ and $\{ab : a \neq b \in \mathbb{A}\}$. An example of a nominal set that is not a renaming set is $\mathbb{A}^{\#n} = \{\, a_1 \dots a_n : a_i \neq a_j \text{ for } i \neq j \,\}$ with pointwise $\mathsf{Perm}(\mathbb{A})$-action.

A nominal set X is *strong* if, for every $x \in X$ and $\pi \in \mathsf{Perm}(\mathbb{A})$, one has $\pi \cdot x = x$ if and only if π fixes every element of $\mathsf{supp}(x)$. (The "if" statement holds in every nominal set.) For instance, the nominal sets $\mathbb{A}^{\#n}$, $\mathbb{A}^n$ and $\mathbb{A}^\star$ are strong. Up to isomorphism, (orbit-finite) strong nominal sets are precisely (finite) coproducts $\coprod_{i\in I} \mathbb{A}^{\#n_i}$ where $n_i \in \mathbb{N}$. For every orbit-finite nominal set X, there exists a surjective $\mathsf{Perm}(\mathbb{A})$-equivariant map $e\colon Y \twoheadrightarrow X$ for some orbit-finite strong nominal set Y (see e.g. [23, Cor. B.27]). In fact, if o is the number of orbits of X, one may take $Y = J \times \mathbb{A}^{\#n}$ where $J = \{1, \dots, o\}$ and $n = \max_{x \in X} |\mathsf{supp}\, x|$. We refer the reader to [14, Sec. 4.1] and [6, Thm. 10.9] for more details on representing orbit-finite nominal sets.

2.2 Nominal Automata and Nominal Renaming Automata

The object of interest in this paper is data languages $L \subseteq \mathbb{A}^\star$ closed under renamings:

▶ **Definition 2.2.**
1. A data language $L \subseteq \mathbb{A}^\star$ is *positive* if it is $\mathsf{Fin}(\mathbb{A})$-equivariant.
2. The *positive closure* of $L \subseteq \mathbb{A}^\star$ is given by $\overline{L} = \{\, \rho^\star(w) : w \in L,\ \rho \in \mathsf{Fin}(\mathbb{A}) \,\}$.

A natural automata model for data languages is given by nondeterministic orbit-finite automata [6] over nominal sets and their restriction to nominal renaming sets:

▶ **Definition 2.3.** Let $M \in \{\, \mathsf{Perm}(\mathbb{A}), \mathsf{Fin}(\mathbb{A}) \,\}$.
1. A *nondeterministic orbit-finite M-automaton* $A = (Q, \delta, I, F)$ consists of an orbit-finite nominal M-set Q of states, an M-equivariant transition relation $\delta \subseteq Q \times \mathbb{A} \times Q$, and M-equivariant subsets $I, F \subseteq Q$ of initial and final states. Nominal orbit-finite M-automata are called *nondeterministic orbit-finite automata* (*NOFA*) for $M = \mathsf{Perm}(\mathbb{A})$ and *nondeterministic orbit-finite renaming automata* (*NOFRA*) for $M = \mathsf{Fin}(\mathbb{A})$.
2. Given a nominal orbit-finite M-automaton A, we write $q \xrightarrow{a} q'$ if $q' \in \delta(q, a)$. A *run* of A on input $w = a_1 \cdots a_n \in \mathbb{A}^\star$ is a sequence $(q_0, a_1, q_1, a_2, \dots, a_n, q_n)$ such that $q_0 \in I$ and $q_r \xrightarrow{a_{r+1}} q_{r+1}$ for $0 \leq r < n$. The run is *accepting* if $q_n \in F$. The automaton A *accepts* the word w if A admits an accepting run on input w. The *accepted language* $L(A) \subseteq \mathbb{A}^\star$ is the set of all accepted words. A data language is *NOF(R)A-recognizable* if some NOF(R)A accepts it.

For example, the languages L_0 and L_1 from the Introduction are NOFA-recognizable.

▶ **Remark 2.4.**
1. The restriction to the input alphabet $\mathbb{A}$ is for simplicity: all our results extend to alphabets $\Sigma = \Sigma_0 \times \mathbb{A}$ for a finite set Σ_0, i.e. finite coproducts of copies of $\mathbb{A}$.
2. Another use of nominal renaming sets in automata theory appears in the work by Moerman and Rot [24] on deterministic nominal automata with outputs. The restrictions of their model make it unsuitable for language recognition [24, Rem. 4.1] but allow for a succinct representation of computed maps via *separating automata*.

To relate the expressive power of NOFA and NOFRA, we start with a simple observation:

▶ **Proposition 2.5.** *Every NOFRA accepts a positive language.*

The converse (Theorem 2.9) needs an automata-theoretic construction of the closure of a language. To this end, we first turn the states of a NOFA into a sort of normal form.

▶ **Remark 2.6** (cf. [6]). Every NOFA $A = (Q, \delta, I, F)$ is equivalent to one whose nominal set of states is of the form $J \times \mathbb{A}^{\#m}$ for some finite set J and $m \in \mathbb{N}$. Indeed, choose a nominal set $Q' = J \times \mathbb{A}^{\#m}$ and an equivariant surjection $e \colon Q' \twoheadrightarrow Q$ (see Section 2.1), and consider the NOFA $A' = (Q', \delta', I', F')$ whose structure is given by the preimages

$$\delta' = (e \times \mathsf{id}_{\mathbb{A}} \times e)^{-1}[\delta], \qquad I' = e^{-1}[I], \qquad F' = e^{-1}[F].$$

It is not difficult to verify that $L(A') = L(A)$; see also Proposition 6.9. Note that in a NOFA with states $J \times \mathbb{A}^{\#m}$, the equivariant sets of initial and final states are of the form $I = J_I \times \mathbb{A}^{\#m}$ and $F = J_F \times \mathbb{A}^{\#m}$ for some $J_I, J_F \subseteq J$.

▶ **Construction 2.7** (Positive Closure of a NOFA)**.** Let $A = (Q, \delta, I, F)$ be a NOFA with states $Q = J \times \mathbb{A}^{\#m}$ (cf. Remark 2.6). The NOFRA $\bar{A} = (\bar{Q}, \bar{\delta}, \bar{I}, \bar{F})$ is given by the states $\bar{Q} = J \times \mathbb{A}^m$, initial states $\bar{I} = J_I \times \mathbb{A}^m$, final states $\bar{F} = J_F \times \mathbb{A}^m$, and transitions

$$\bar{\delta} \;=\; \{\, (j, \rho^\star p) \xrightarrow{\rho a} (j', \rho^\star p') : (j,p) \xrightarrow{a} (j',p') \text{ in } A \text{ and } \rho \in \mathsf{Fin}(\mathbb{A}) \,\}.$$

▶ **Proposition 2.8.** *The NOFRA $\bar{A}$ accepts the positive closure of the language of A.*

The proof of $L(\bar{A}) \subseteq \overline{L(A)}$ is slightly subtle since the transitions of a run in $\bar{A}$ may be induced by different ρ's; some bookkeeping and sensible choice of fresh names ensures compatibility.

Now we come to our first characterization of positive NOFA-recognizable languages:

▶ **Theorem 2.9.** *A language is positive and NOFA-recognizable iff it is NOFRA-recognizable.*

Indeed, the "if" direction holds due to Proposition 2.5 and because every NOFRA is a NOFA. The "only if" direction follows from Proposition 2.8, using that $\overline{L} = L$ for positive L.

▶ **Remark 2.10.** A NOF(R)A is *deterministic*, and hence called a *DOF(R)A*, if it admits a single initial state and its transition relation is a function $\delta\colon Q \times \mathbb{A} \to Q$. In contrast to classical finite automata, DOFAs are less expressive that NOFAs [6]. We leave it as an open problem whether Theorem 2.9 restricts to DOF(R)As. In this regard, observe that Construction 2.7 produces a *non*deterministic automaton $\bar{A}$ even if the given automaton A is deterministic. Computing the positive closure of a DOFA-recognizable language necessarily requires the introduction of nondeterminism, as illustrated by the following example due to Bartek Klin (personal communication). Consider the language L consisting of all words whose last letter appears immediately before the last occurrence of a repeated letter; that is, words of the form $vabbwa$ where (i) $v, w \in \mathbb{A}^\star$ and $a, b \in \mathbb{A}$, (ii) any two consecutive letters in w are distinct, (iii) the first letter of w is distinct from b and (iv) the last letter of w is distinct from a. This language is recognizable by a DOFA, in fact by an orbit-finite nominal monoid [4]. Its positive closure $\overline{L}$ consists of all words whose last letter appears immediately before *some* occurrence of a repeated letter, which is not DOFA-recognizable.

2.3 Abstract Transitions and Runs

Sections 3 and 4 will relate positive NOFA-recognizable languages to register automata and monadic second-order logic. This relies on a presentation of transitions of $\bar{A}$ in terms of abstract transitions, given by equations involving register entries and input values.

▶ **Definition 2.11.** Let $A = (Q, \delta, I, F)$ and $\bar{A} = (\bar{Q}, \bar{\delta}, \bar{I}, \bar{F})$ be as in Construction 2.7.

1. An *equation* is an expression of the form $k = \bullet$, $\bullet = k$ or $k = \bar{k}$, where $k, \bar{k} \in \{1, \ldots, m\}$.
2. An *abstract transition* is a triple (j, E, j') where $j, j' \in J$ and E is a set of equations.
3. Every triple $((j,p), a, (j',p')) \in Q \times \mathbb{A} \times Q$ induces an abstract transition (j, E, j') defined as follows for $k, \bar{k} \in \{1, \ldots, m\}$ (we write $(-)_i$ for the i-th letter of a word):
$$k = \bullet \in E \iff p_k = a, \qquad \bullet = k \in E \iff a = p'_k, \qquad k = \bar{k} \in E \iff p_k = p'_{\bar{k}}.$$
We let $\mathsf{abs}(\delta)$ denote the set of abstract transitions induced by transitions in δ, and we write $j \xrightarrow{E} j'$ for $(j, E, j') \in \mathsf{abs}(\delta)$.

4. A triple $((j,q),b,(j',q')) \in \bar{Q} \times \mathbb{A} \times \bar{Q}$ is *consistent* with the abstract transition (j,E,j') if for every $k,\bar{k} \in \{1,\dots,m\}$ the following conditions hold:

$$k = \bullet \in E \implies q_k = b, \qquad \bullet = k \in E \implies b = q'_k, \qquad k = \bar{k} \in E \implies q_k = q'_{\bar{k}}.$$

▶ **Proposition 2.12.** *For every triple $((j,q),b,(j',q')) \in \bar{Q} \times \mathbb{A} \times \bar{Q}$, we have*

$$(j,q) \xrightarrow{b} (j',q') \text{ in } \bar{A} \quad \text{iff} \quad ((j,q),b,(j',q')) \text{ is consistent with some } (j,E,j') \in \mathsf{abs}(\delta).$$

▶ **Definition 2.13.** An *abstract run* in $\bar{A}$ is a sequence $(j_0,E_1,j_1,E_2,j_2,\dots,E_n,j_n)$ such that $j_0 \in J_I$ and $j_{r-1} \xrightarrow{E_r} j_r$ for $r = 1,\dots,n$. It is *accepting* if $j_n \in J_F$.

▶ **Notation 2.14.** Given an abstract run $(j_0,E_1,j_1,E_2,j_2,\dots,E_n,j_n)$, we inductively define the predicates $\mathsf{Eq}_k^{(i)}$ ($i \in \{1,\dots,n\}$, $k \in \{1,\dots,m\}$) on the set $\{1,\dots,n\}$:
1. if $\bullet = k$ in E_i then $\mathsf{Eq}_k^{(i)}(i)$;
2. if $r < n$ and $k = \bar{k}$ in E_{r+1} and $\mathsf{Eq}_k^{(i)}(r)$ then $\mathsf{Eq}_{\bar{k}}^{(i)}(r+1)$.

Informally, $\mathsf{Eq}_k^{(i)}(r)$ asserts that $1 \le i \le r \le n$ and that in every run in $\bar{A}$ of length r whose transitions are consistent with $E_1,\dots,E_r$, the i-th input letter equals the content of register k after r steps. The accepted language may be characterized using these predicates:

▶ **Proposition 2.15.** *The NOFRA $\bar{A}$ accepts the word $b_1 \cdots b_n \in \mathbb{A}^\star$ iff there exists an accepting abstract run of length n (with induced predicates $\mathsf{Eq}_k^{(i)}$) such that for $i,r \in \{1,\dots,n\}$,*

$$r < n \text{ and } k = \bullet \text{ in } E_{r+1} \text{ and } \mathsf{Eq}_k^{(i)}(r) \text{ for some } k \quad \implies \quad b_i = b_{r+1}. \tag{2.1}$$

As a first application of this result, we identify an important difference between NOFA and NOFRA concerning the power of guessing data values during the computation:

▶ **Definition 2.16.** A NOFA/NOFRA is *non-guessing* if each initial state has empty support and for each transition $q \xrightarrow{a} q'$ one has $\mathsf{supp}\, q' \subseteq \mathsf{supp}\, q \cup \{a\}$.

The NOFA-recognizable language L_0 from the Introduction is not recognizable by any non-guessing NOFA [17, Ex. 1]. Note that L_0 is not positive; in fact, it is necessarily so, since for positive languages guessing does not add to the expressive power of automata:

▶ **Theorem 2.17.** *Every positive NOFA-recognizable language is accepted by some non-guessing NOFRA, hence by some non-guessing NOFA.*

To make a NOFRA non-guessing, one keeps track (via the state) of those registers containing data values forced by abstract transitions. The other registers then may be modified arbitrarily, which allows the elimination of guessing transitions.

3 Positive Register Automata

We now relate positive NOFA-recognizable languages to register automata, a.k.a. finite-memory automata, originally introduced by Kaminski and Francez [17]; we follow the equivalent presentation by Bojańczyk et al. [6]. A *register automaton* is a quintuple $A = (C,m,\delta,I,F)$ where C is a finite set of control states, $m \in \mathbb{N}$ is the number of registers (numbered from 1 to m), $I,F \subseteq C$ are sets of initial and final states, and $\delta \subseteq C \times \mathrm{Bool}(\Phi) \times C$ is the set of transitions. Here, $\mathrm{Bool}(\Phi)$ denotes the set of boolean formulas over the atoms $\Phi = (\{1,\dots,m\} \times \{\text{before}\} \cup \{\bullet\} \cup \{1,\dots,m\} \times \{\text{after}\})^2$. Elements of Φ are called *equations*; we write $x = y$ for $(x,y) \in \Phi$. Moreover, we denote $(c,\varphi,c') \in \delta$ by $c \xrightarrow{\varphi} c'$. A *configuration* of A is a pair (c, r) of a state $c \in C$ and a word $r \in (\mathbb{A} \cup \{\bot\})^m$ corresponding to a partial assignment of data values to the registers. The initial configurations are $(c, \bot^m)$ for $c \in I$. Given an input $a \in \mathbb{A}$ and configurations $(c, r), (c', r')$ we write $(c, r) \xrightarrow{a} (c', r')$ if this move is consistent with some transition $c \xrightarrow{\varphi} c'$, that is, the formula φ is true under the assignment making an atom $x = y \in \Phi$ true iff the corresponding data values are defined and equal. For instance, $(k, \text{before}) = \bullet$ is true iff $r_k \neq \bot$ and $r_k = a$, and $(k, \text{before}) = (\bar{k}, \text{after})$ is true iff $r_k, r'_{\bar{k}} \neq \bot$ and $r_k = r'_{\bar{k}}$. A word $w = a_1 \ldots a_n \in \mathbb{A}^\star$ is *accepted* by A if it admits an accepting run, viz. a sequence of moves $(c_0, r_0) \xrightarrow{a_1} (c_1, r_1) \xrightarrow{a_2} \cdots \xrightarrow{a_n} (c_n, r_n)$ where (c_0, r_0) is initial and $c_n \in F$. The *accepted language* $L(A) \subseteq \mathbb{A}^\star$ is the set of accepted words.

As shown by Bojańczyk et al. [6], register automata accept the same languages as NOFAs. To capture positive languages, we restrict to register automata with positive transitions:

▶ **Definition 3.1.** A register automaton is *positive* if for each transition $c \xrightarrow{\varphi} c'$ the formula φ is positive: $\varphi = \text{true}$ or φ uses the boolean operations $\vee$ and $\wedge$ only.

▶ **Theorem 3.2.** *A data language is positive and NOFA-recognizable iff it is accepted by some positive register automaton.*

Here, the approach is to regard a configuration of a positive register automaton as a state of a NOFRA. Conversely, an abstract transition $j \xrightarrow{E} j'$ of a NOFA can be transformed into a transition $j \xrightarrow{\varphi} j'$ of a register automaton for the conjunction φ of all equations in E, identifying $k = \bullet$, $\bullet = k$, $k = \bar{k}$ with $(k, \text{before}) = \bullet$, $\bullet = (k, \text{after})$, $(k, \text{before}) = (\bar{k}, \text{after})$. A tweak of the initial states accounts for the requirement that registers are initially empty.

▶ **Remark 3.3.** Just like register automata are equivalent to finite-memory automata, positive register automata correspond to a restricted version of finite-memory automata called *finite-state unification-based automata* (*FSUBA*) [18,32]. The original definition of the latter involves a fixed initial register assignment, which enables acceptance of non-positive languages. However, FSUBA with empty initial registers are equivalent to positive register automata; see the full paper for details. This implies in particular that positive register automata admit a decidable inclusion problem, in contrast to the case of unrestricted register automata [25]. Indeed, FSUBA translate into a more general model called *RNNA* [29, Sec. 6], for which inclusion is decidable. Tal [32] has given a direct decidability proof for FSUBA.

4 Monadic Second-Order Logic with Positive Equality Tests

As motivated in the Introduction, positive data languages are considered as expressing properties of data words involving positive statements about equality of data values. In the following we make this idea precise. For this purpose, we employ monadic second-order logic with equality tests, abbreviated MSO$^\sim$ [4,9,25]. Its formulae are given by the grammar

$$\varphi, \psi \quad ::= \quad x < y \mid x \sim y \mid X(x) \mid \neg\varphi \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \exists x.\, \varphi \mid \exists X.\, \varphi \mid \forall x.\, \varphi \mid \forall X.\, \varphi,$$

where x, y range over first-order variables and X over monadic second-order variables. A formula is interpreted over a fixed data word $w = a_1 \ldots a_n \in \mathbb{A}^\star$. First-order variables represent positions, i.e. elements of the set $\{1, \ldots, n\}$, and second-order variables represent subsets of $\{1, \ldots, n\}$. The atomic formula $x < y$ means "position x comes before position y", and $x \sim y$ means "the same data value occurs at positions x and y". The interpretation of the remaining constructs is standard. A *sentence* is a formula without free variables. We write $L(\varphi) \subseteq \mathbb{A}^\star$ for the set of data words satisfying the sentence φ. For example, the languages L_0 and L_1 from the Introduction are defined by $\varphi_0 = \forall y.\, \text{last}(y) \Rightarrow (\forall x.\, x < y \Rightarrow \neg(x \sim y))$, where $\text{last}(y) = \neg\exists z.\, y < z$ and $\psi \Rightarrow \xi = \neg\psi \vee \xi$, and by $\varphi_1 = \exists x.\, \exists y.\, x < y \wedge x \sim y$.

Recall that by standard rules of negation, every formula is equivalent to one in *negation normal form* (*NNF*), where for each subformula $\neg\varphi$ the formula φ is atomic.

▶ **Definition 4.1.** An MSO$^{\sim}$ formula lies in MSO$^{\sim,+}$ (*monadic second-order logic with positive equality tests*) if it admits an NNF containing no subformula of the form $\neg(x \sim y)$. A data language is *MSO$^{\sim,+}$-definable* if it is of the form $L(\varphi)$ for an MSO$^{\sim,+}$ sentence φ.

The above sentence φ_1 lies in MSO$^{\sim,+}$ but φ_0 does not. The following is immediate:

▶ **Proposition 4.2.** *Every MSO$^{\sim,+}$-definable language is positive.*

▶ **Remark 4.3.** The logic MSO$^{\sim}$ is more expressive than NOFAs [25], and the same holds for MSO$^{\sim,+}$: the language defined by the MSO$^{\sim,+}$ sentence $\varphi = \forall x.\, \exists y.\, (x < y \vee y < x) \wedge x \sim y$ ("no data value occurs only once") is not NOFA-recognizable. However, within the class of NOFA-recognizable languages, positive and MSO$^{\sim,+}$-definable languages coincide:

▶ **Theorem 4.4.** *A NOFA-recognizable language is positive iff it is MSO$^{\sim,+}$-definable.*

Indeed, one can express the abstract acceptance condition of Proposition 2.15 in MSO$^{\sim,+}$.

5 Toposes for Names

In the remainder, we investigate positive data languages and their automata from a more conceptual perspective. Some familiarity with basic category theory (functors, natural transformations, (co-)limits, adjunctions) is required; see Mac Lane [22] for a gentle introduction.

Nominal sets and nominal renamings sets (Section 2.1) were initially introduced as a convenient abstract framework for reasoning about names, and related issues such as freshness, binding, and substitution. An alternative, and more general, approach uses the presheaf categories $\mathbf{Set}^{\mathbb{I}}$ [31] and $\mathbf{Set}^{\mathbb{F}}$ [10]. The intuition behind each of these categories $\mathscr{C}$ is very similar: one thinks of $X \in \mathscr{C}$ as a collection of finitely supported objects, equipped with a renaming operation that extends renamings $\rho\colon \mathbb{A} \to \mathbb{A}$ to the level of elements of X. The difference between the four categories lies in whether elements admit a *least* support, or just some finite support, and in whether renamings ρ are injective or arbitrary maps; see Figure 2. The last column classifies the respective finitely presentable objects, which underly automata. We now recall the latter concept and describe the categories in more detail.

Finitely presentable objects. A diagram $D\colon I \to \mathscr{C}$ in a category $\mathscr{C}$ is *directed* if its scheme I is a directed poset: every finite subset of I has an upper bound. A *directed colimit* is a colimit of a directed diagram. An object X of $\mathscr{C}$ is *finitely presentable* if its hom-functor $\mathscr{C}(X,-)\colon \mathscr{C} \to \mathbf{Set}$ to the category of sets and functions preserves directed colimits. In many categories, finitely presentable objects correspond to the objects with a finite description. For example, the finitely presentable objects of $\mathbf{Set}$ are precisely finite sets, and if $\mathscr{C}$ is a variety of algebras (e.g. monoids, groups, rings), an algebra is a finitely presentable object of $\mathscr{C}$ iff it is presentable by finitely many generators and relations [2, Thm. 3.12].

Nominal (renaming) sets. We let $\mathbf{Nom}$ denote the category of nominal sets and $\mathsf{Perm}(\mathbb{A})$-equivariant maps, and $\mathbf{RnNom}$ the category of nominal renaming sets and $\mathsf{Fin}(\mathbb{A})$-equivariant maps. Both categories are toposes, that is, they are finitely complete (with limits formed as in $\mathbf{Set}$), cartesian closed, and admit a subobject classifier. Note that $\mathbf{Nom}$ is a boolean topos (its subobject classifier is $2 = \{0,1\}$ with the trivial group action), which is not true for $\mathbf{RnNom}$ [12, Sec. 5]. The next proposition provides a categorical description of orbit-finite nominal (renaming) sets; for nominal sets this result is well-known, see [26, Prop. 2.3.7] or [27, Thm. 5.16], and the statement for nominal renaming sets may be deduced from it.

Category	Objects	Least supp.	Renamings	Finitely pres. objects
Nom	nominal sets	yes	injective	orbit-finite sets
RnNom	nominal renaming sets	yes	arbitrary	orbit-finite sets
$\mathbf{Set}^{\mathbb{I}}$	presheaves over $\mathbb{I}$	no	injective	super-finitary presheaves
$\mathbf{Set}^{\mathbb{F}}$	presheaves over $\mathbb{F}$	no	arbitrary	super-finitary presheaves

Figure 2 Toposes that model sets of finitely supported objects.

▶ **Proposition 5.1.** *A nominal (renaming) set is orbit-finite iff it is a finitely presentable object of* **Nom** *or* **RnNom**, *respectively.*

The forgetful functor $U\colon \mathbf{RnNom} \to \mathbf{Nom}$ given by restricting the $\mathsf{Fin}(\mathbb{A})$- to a $\mathsf{Perm}(\mathbb{A})$-action has a left adjoint $F\colon \mathbf{Nom} \to \mathbf{RnNom}$ [24, Thm. 2.6]. We refer to *op. cit.* for its explicit description, but remark that $F(\mathbb{A}^{\#n}) = \mathbb{A}^n$ for every $n \in \mathbb{N}$ [24, Thm. 3.7].

Presheaves. A *(covariant) presheaf* over a small category $\mathscr{C}$ is a functor $P\colon \mathscr{C} \to \mathbf{Set}$. We write $\mathbf{Set}^{\mathscr{C}}$ for the category of presheaves and natural transformations. We specifically consider presheaves over $\mathbb{F}$ and $\mathbb{I}$, the categories whose objects are finite subsets $S \subseteq_{\mathsf{f}} \mathbb{A}$ and whose morphisms $\rho\colon S \to T$ are functions or injective functions, respectively. The categories **Nom** and **RnNom** form full reflective subcategories of $\mathbf{Set}^{\mathbb{I}}$ and $\mathbf{Set}^{\mathbb{F}}$ via embeddings

$$I_\star\colon \mathbf{Nom} \rightarrowtail \mathbf{Set}^{\mathbb{I}} \qquad \text{and} \qquad J_\star\colon \mathbf{RnNom} \rightarrowtail \mathbf{Set}^{\mathbb{F}}.$$

Here, $I_\star$ is given for $X \in \mathbf{Nom}$, $S \subseteq_{\mathsf{f}} \mathbb{A}$, $\rho\colon S \to T$ in $\mathbb{I}$ and $f\colon X \to Y$ in **Nom** by

$$(I_\star X)S = \{\, x \in X : \mathsf{supp}\, x \subseteq S \,\}, \qquad (I_\star X)\rho(x) = \overline{\rho} \cdot x, \qquad (I_\star f)_S(x) = f(x),$$

where $\overline{\rho} \in \mathsf{Perm}(\mathbb{A})$ is any permutation extending the injective map ρ. The embedding $J_\star$ is defined analogously. In both cases, the essential image of the embedding consists precisely of the presheaves preserving pullbacks of injective maps, see [27, Thm. 6.8] and [12, Thm. 38]. Informally, a presheaf $P \in \mathbf{Set}^{\mathscr{C}}$, where $\mathscr{C} \in \{\mathbb{I}, \mathbb{F}\}$, specifies a set PS of S-supported objects for every $S \subseteq_{\mathsf{f}} \mathbb{A}$, and the pullback preservation property asserts precisely that every object admits a least support. A presheaf $P \in \mathbf{Set}^{\mathscr{C}}$ is *super-finitary* if there exists $S \subseteq_{\mathsf{f}} \mathbb{A}$ such that (i) PS' is a finite set for all $S' \subseteq S$, and (ii) for every $T \subseteq_{\mathsf{f}} \mathbb{A}$ and $x \in PT$, there exists $S' \subseteq S$ and $\rho \in \mathscr{C}(S', T)$ such that $x \in P\rho[PS']$. (This implies that PT is finite.) Such an S is called a *generating set* for P. The next proposition shows that super-finitary presheaves are the analogue of orbit-finite sets; see [1, Cor. 3.34] for the case $\mathscr{C} = \mathbb{F}$:

▶ **Proposition 5.2.** *For $\mathscr{C} \in \{\mathbb{I}, \mathbb{F}\}$ and $P \in \mathbf{Set}^{\mathscr{C}}$, the following are equivalent: (i) P is super-finitary; (ii) P is finitely presentable; (iii) there exists an epimorphism (a componentwise surjective natural transformation) $\coprod_{i \in I} \mathscr{C}(S_i, -) \twoheadrightarrow P$ with I finite and $S_i \subseteq_{\mathsf{f}} \mathbb{A}$. Moreover, super-finitary presheaves are closed under sub-presheaves and finite products.*

To relate the two presheaf categories $\mathbf{Set}^{\mathbb{I}}$ and $\mathbf{Set}^{\mathbb{F}}$, recall that every functor $E\colon \mathscr{C} \to \mathscr{D}$ between small categories induces an adjunction (5.1), where the right adjoint $E^\star$ is given by $E^\star(P) = P \circ E$, and the left adjoint sends a presheaf $P \in \mathbf{Set}^{\mathscr{C}}$ to its *left Kan extension* $\mathsf{Lan}_E P$. For the inclusion functor $E\colon \mathbb{I} \hookrightarrow \mathbb{F}$, we obtain the commutative diagram (5.2) of adjunctions. Here, $I^\star$ and $J^\star$ are the reflectors, i.e. the left adjoints of $I_\star$ and $J_\star$.

$$\begin{tikzcd}
\mathbf{Set}^{\mathscr{C}} \arrow[r, bend right, "\mathsf{Lan}_E"'] \arrow[r, phantom, "\top"] & \mathbf{Set}^{\mathscr{D}} \arrow[l, bend right, "E^\star"']
\end{tikzcd} \tag{5.1}$$

$$\begin{tikzcd}
\mathbf{Set}^{\mathbb{I}} \arrow[r, bend right, "\mathsf{Lan}_E"'] \arrow[r, phantom, "\top"] \arrow[d, bend left, "I^\star"] \arrow[d, phantom, "\vdash"] & \mathbf{Set}^{\mathbb{F}} \arrow[l, bend right, "E^\star"'] \arrow[d, bend right, "J^\star"'] \arrow[d, phantom, "\dashv"] \\
\mathbf{Nom} \arrow[u, bend left, "I_\star"] \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & \mathbf{RnNom} \arrow[l, bend left, "U"] \arrow[u, bend right, "J_\star"']
\end{tikzcd} \tag{5.2}$$

▶ **Proposition 5.3.** *All functors in* (5.2) *preserve finitely presentable objects.*

Hence, the adjunctions (5.2) restrict to the full subcategories of finitely presentable objects.

6 Nondeterministic Automata in a Category

Our aim is to investigate nondeterministic automata and their languages in the toposes of Figure 2, and to compare their expressive power. To this end, we first introduce the required automata-theoretic concepts uniformly at the level of abstract categories.

▶ **Assumptions 6.1.** Fix a category $\mathscr{C}$ with finite limits and (strong epi, mono)-factorizations. We assume that strong epimorphisms are stable under finite products (that is, $e \times e'$ is a strong epimorphism whenever e and e' are) and pullbacks (that is, in every pullback square $e \circ \overline{f} = f \circ \overline{e}$, the morphism $\overline{e}$ is a strong epimorphism whenever e is).

The (strong epi, mono)-factorization $f = (A \xrightarrow{e} I \xrightarrow{m} B)$ of a morphism $f\colon A \to B$ in $\mathscr{C}$ is its *image factorization*, and the subobject represented by m is the *image* of f.

▶ **Example 6.2.** Every topos satisfies Assumptions 6.1, including **Set**, **Nom**, **RnNom**, $\mathbf{Set}^{\mathbb{I}}$ and $\mathbf{Set}^{\mathbb{F}}$. Note that in a topos all epimorphisms are strong. In the five categories above, epi- and monomorphisms are the (componentwise) surjective and injective morphisms, resp. Pullbacks and finite products are formed (componentwise) at the level of underlying sets.

▶ **Definition 6.3.** A *language* over $\Sigma \in \mathscr{C}$ is a family of subobjects of Σ^n for each $n \in \mathbb{N}$:

$$L \;=\; (\, m_n^{(L)} \colon L^{(n)} \rightarrowtail \Sigma^n \,)_{n \in \mathbb{N}}.$$

We write $L \leq L'$ iff $L^{(n)} \leq L'^{(n)}$ for all n, using the partial order $\leq$ on subobjects of Σ^n.

▶ **Remark 6.4.** If $\mathscr{C}$ is countably extensive (e.g. a topos with countable coproducts), languages correspond bijectively to subobjects of $\Sigma^\star = \coprod_{n \in \mathbb{N}} \Sigma^n$. Indeed, every language L yields the subobject $\coprod_n m_n^{(L)} \colon \coprod_n L^{(n)} \rightarrowtail \Sigma^\star$, and conversely every subobject of $\Sigma^\star$ is of this form. In particular, this holds in the categories of Example 6.2.

▶ **Definition 6.5.** A *nondeterministic $\mathscr{C}$-automaton* is a quintuple $A = (Q, \Sigma, \delta, I, F)$ consisting of an object $Q \in \mathscr{C}$ of *states*, an *input alphabet* $\Sigma \in \mathscr{C}$, and subobjects

$$m_\delta \colon \delta \rightarrowtail Q \times \Sigma \times Q, \qquad m_I \colon I \rightarrowtail Q, \qquad m_F \colon F \rightarrowtail Q,$$

representing *transitions*, *initial states*, and *final states*, respectively. A *morphism* $h\colon A' \to A$ of nondeterministic $\mathscr{C}$-automata is given by a pair of morphisms $h_{\mathsf{s}} \colon Q' \to Q$ and $h_{\mathsf{a}} \colon \Sigma' \to \Sigma$ of $\mathscr{C}$ that restrict as shown below (note that h_{t}, h_{i} and h_{f} are uniquely determined):

$$\begin{tikzcd}
\delta' \arrow[r, dashed, "h_{\mathsf{t}}"] \arrow[d, tail, "m_{\delta'}"'] & \delta \arrow[d, tail, "m_\delta"] & I' \arrow[r, dashed, "h_{\mathsf{i}}"] \arrow[d, tail, "m_{I'}"'] & I \arrow[d, tail, "m_I"] & F' \arrow[r, dashed, "h_{\mathsf{f}}"] \arrow[d, tail, "m_{F'}"'] & F \arrow[d, tail, "m_F"] \\
Q' \times \Sigma' \times Q' \arrow[r, "h_{\mathsf{s}} \times h_{\mathsf{a}} \times h_{\mathsf{s}}"] & Q \times \Sigma \times Q & Q' \arrow[r, "h_{\mathsf{s}}"] & Q & Q' \arrow[r, "h_{\mathsf{s}}"] & Q
\end{tikzcd} \tag{6.1}$$

We write $\mathbf{NAut}(\mathscr{C})$ for the category of nondeterministic automata in $\mathscr{C}$ and their morphisms, and $\mathbf{NAut}_{\mathsf{fp}}(\mathscr{C})$ for its full subcategory given by *nondeterministic fp-automata*, viz. automata where Q, Σ, δ, I, F are finitely presentable objects of $\mathscr{C}$.

▶ **Definition 6.6.** For every nondeterministic $\mathscr{C}$-automaton $A = (Q, \Sigma, \delta, I, F)$, its *accepted language* is the language $L(A)$ over Σ given as follows:

1. $m_{L(A)}^{(0)} \colon L^{(0)}(A) \rightarrowtail 1 = \Sigma^0$ is the image of the unique morphism $I \cap F \xrightarrow{!} 1$, where 1 is the terminal object of $\mathscr{C}$ and $I \cap F$ is the intersection (pullback) of m_I and m_F.

2. For $n > 0$, the subobject $m^{(n)}_{L(A)} \colon L^{(n)}(A) \rightarrowtail \Sigma^n$ is defined via the commutative diagram

$$\begin{tikzcd}
L^{(n)}(A) \arrow[d, tail, "m^{(n)}_{L(A)}"'] & \mathsf{AccRun}^{(n)}_A \arrow[l, two heads, "e_{n,A}"'] \arrow[r, tail, "\overline{d}_{n,A}"] \arrow[d, tail, "\overline{m}^{(n)}_\delta"] \arrow[dr, phantom, "\lrcorner", very near start] & \delta^n \arrow[d, tail, "m^n_\delta"] \\
\Sigma^n & I \times (\Sigma \times Q)^{n-1} \times \Sigma \times F \arrow[l, "p_{n,A}"'] \arrow[r, tail, "d_{n,A}"] & (Q \times \Sigma \times Q)^n
\end{tikzcd}$$

Here, letting $\Delta \colon Q \rightarrowtail Q \times Q$ denote the diagonal, $d_{n,A}$ is the monomorphism

$$I \times (\Sigma \times Q)^{n-1} \times \Sigma \times F \xrightarrow{m_I \times (\mathsf{id} \times \Delta)^{n-1} \times \mathsf{id} \times m_F} Q \times (\Sigma \times Q \times Q)^{n-1} \times \Sigma \times Q \cong (Q \times \Sigma \times Q)^n,$$

the morphisms $\overline{d}_{n,A}$ and $\overline{m}^{(n)}_\delta$ form the pullback of $d_{n,A}$ and m^n_δ, the morphism $p_{n,A}$ is the projection, and $e_{n,A}$ and $m^{(n)}_{L(A)}$ form the image factorization of $p_{n,A} \circ \overline{m}^{(n)}_\delta$.

▶ **Example 6.7.**
1. A nondeterministic fp-automaton in **Set** is a classical nondeterministic finite automaton. The pullback $\mathsf{AccRun}^{(n)}_A$ is the set of accepting runs of length n, hence $L(A)$ is the usual accepted language: the set of words with an accepting run.
2. A nondeterministic fp-automaton in **Nom** or **RnNom** with alphabet $\Sigma = \mathbb{A}$ is a NOFA or NOFRA, respectively. The two notions of accepted language in Definition 2.3 and Definition 6.6 match, that is, $L(A)$ is the set of words with an accepting run.
3. In the next section, we will also look into nondeterministic $\mathbf{Set}^{\mathbb{I}}$- and $\mathbf{Set}^{\mathbb{F}}$-automata.

▶ **Remark 6.8.** Readers familiar with coalgebras [28] may note that if $\mathscr{C}$ is a topos, the final states and transitions of a nondeterministic $\mathscr{C}$-automaton correspond to a coalgebra $\gamma \colon Q \to \Omega \times (\mathcal{P}Q)^\Sigma$ where Ω is the subobject classifier and $\mathcal{P} \colon \mathscr{C} \to \mathscr{C}$ is the covariant power object functor [16, Sec. A.2.3]. We expect our above definition of accepted language to match the one given by coalgebraic trace semantics [15,30], with the required arguments relying on the internal logic of the topos $\mathscr{C}$. Details are left for future work; we have found that the present relational approach to automata leads to shorter and more direct proofs.

▶ **Proposition 6.9.** *Let $h \colon A' \to A$ be an $\mathbf{NAut}(\mathscr{C})$-morphism where $\Sigma' = \Sigma$ and $h_{\mathsf{a}} = \mathsf{id}_\Sigma$.*
1. *The accepted language of A' is contained in that of A, that is, $L(A') \leq L(A)$.*
2. *If h_{s} is strongly epic in $\mathscr{C}$ and the squares (6.1) are pullbacks, then $L(A') = L(A)$.*

Hence, the construction $A \mapsto A'$ of Remark 2.6 indeed yields an equivalent NOFA.

▶ **Proposition 6.10.** *Let $\mathscr{C}$ and $\mathscr{D}$ be categories satisfying the Assumptions 6.1.*
1. *Every functor $G \colon \mathscr{C} \to \mathscr{D}$ lifts to a functor $\bar{G} \colon \mathbf{NAut}(\mathscr{C}) \to \mathbf{NAut}(\mathscr{D})$ defined by*

$$\bar{G}(Q, \Sigma, \delta, I, F) = (GQ, G\Sigma, \overline{G\delta}, \overline{GI}, \overline{GF}) \qquad \textit{and} \qquad \bar{G}f = Gf.$$

Here, $\overline{G\delta}$, $\overline{GI}$, $\overline{GF}$ are the images of the morphisms shown below, with can *denoting the canonical morphism induced by the product projections:*

$$G\delta \xrightarrow{Gm_\delta} G(Q \times \Sigma \times Q) \xrightarrow{\mathsf{can}} GQ \times G\Sigma \times GQ, \qquad GI \xrightarrow{Gm_I} GQ, \qquad GF \xrightarrow{Gm_F} GQ.$$

2. *Every adjunction $L \dashv R \colon \mathscr{C} \to \mathscr{D}$ lifts to an adjunction $\overline{L} \dashv \overline{R} \colon \mathbf{NAut}(\mathscr{C}) \to \mathbf{NAut}(\mathscr{D})$.*

In particular, the adjunctions (5.2) lift to adjunctions between the respective categories of nondeterministic automata, which in turn restrict to fp-automata by Proposition 5.3:

$$\begin{tikzcd}
\mathbf{NAut}_{\mathsf{fp}}(\mathbf{Set}^{\mathbb{I}}) \arrow[r, shift right, "\mathsf{Lan}_E"'] \arrow[r, phantom, "\top"] \arrow[d, shift right, "\bar{I}^\star"'] \arrow[d, phantom, "\vdash"] & \mathbf{NAut}_{\mathsf{fp}}(\mathbf{Set}^{\mathbb{F}}) \arrow[l, shift right, "\bar{E}^\star"'] \arrow[d, shift left, "\bar{J}^\star"] \arrow[d, phantom, "\dashv"] \\
\mathbf{NAut}_{\mathsf{fp}}(\mathbf{Nom}) \arrow[r, shift left, "\bar{F}"] \arrow[r, phantom, "\perp"] \arrow[u, shift right, "\bar{I}_\star"'] & \mathbf{NAut}_{\mathsf{fp}}(\mathbf{RnNom}) \arrow[l, shift left, "\bar{U}"] \arrow[u, shift left, "\bar{J}_\star"]
\end{tikzcd} \tag{6.2}$$

The positive closure $A \mapsto \bar{A}$ of Construction 2.7, which is key to our results in Sections 2 through 4, is an instance of the proposition since $\bar{A} = \bar{F}A$ for the left adjoint $F\colon \mathbf{Nom} \to \mathbf{RnNom}$.

7 Nondeterministic Presheaf Automata

We proceed to relate the expressive power of the four automata models in (6.2). Specifically, for $\mathscr{C} \in \{\mathbb{I}, \mathbb{F}\}$ we consider nondeterministic $\mathbf{Set}^{\mathscr{C}}$-automata $A = (Q, \Sigma, \delta, I, F)$ with a super-finitary (= finitely presentable) presheaf Q of states and input alphabet $\Sigma = V_{\mathscr{C}} \in \mathbf{Set}^{\mathscr{C}}$, for the inclusion functor $V_{\mathscr{C}}(S) = S$. (This implies that δ, I and F are super-finitary by Proposition 5.2.) Note that $V_{\mathscr{C}}$ corresponds to the input alphabet $\mathbb{A}$ used for NOF(R)As:

$$V_{\mathbb{I}} = I_\star(\mathbb{A}) \qquad \text{and} \qquad V_{\mathbb{F}} = J_\star(\mathbb{A}) = \mathsf{Lan}_E(V_{\mathbb{I}}).$$

A *language* in $\mathbf{Set}^{\mathscr{C}}$ is a sub-presheaf $L \subseteq V_{\mathscr{C}}^\star$, or equivalently a family of sub-presheaves $L^{(n)} \subseteq V_{\mathscr{C}}^n$ for $n \in \mathbb{N}$ (Definition 6.3 and Remark 6.4). Here, $V_{\mathscr{C}}^\star(S) = S^\star$, the set of words over the finite alphabet $S \subseteq_{\mathsf{f}} \mathbb{A}$, and $V_{\mathscr{C}}^n(S) = S^n$, the subset of words of length n.

▶ **Remark 7.1.** For the sake of distinction, we refer to languages in $\mathbf{Set}^{\mathscr{C}}$ as *presheaf languages*, and to subsets of $\mathbb{A}^\star$ as *word languages*. Both concepts are closely related: Every presheaf language $L \subseteq V_{\mathbb{I}}^\star$ in $\mathbf{Set}^{\mathbb{I}}$ induces a $\mathsf{Perm}(\mathbb{A})$-equivariant word language $\mathsf{W}(L) \subseteq \mathbb{A}^\star$ given by $\mathsf{W}(L) = \bigcup_{S \subseteq_{\mathsf{f}} \mathbb{A}} L(S)$, and, conversely, every $\mathsf{Perm}(\mathbb{A})$-equivariant word language $K \subseteq \mathbb{A}^\star$ induces a presheaf language $\mathsf{P}(K) \subseteq V_{\mathbb{I}}^\star$ given by $[\mathsf{P}(K)]S = K \cap S^\star$ for $S \subseteq_{\mathsf{f}} \mathbb{A}$. Analogously for presheaf languages in $\mathbf{Set}^{\mathbb{F}}$ and $\mathsf{Fin}(\mathbb{A})$-equivariant word languages. In both cases, these translations almost yield a bijective correspondence: one has $K = \mathsf{W}(\mathsf{P}(K))$, but generally only $L \subseteq \mathsf{P}(\mathsf{W}(L))$. For instance, for $L \subseteq V_{\mathbb{F}}^\star$ given by $L(\emptyset) = \emptyset$ and $L(S) = \{\varepsilon\}$ for $S \neq \emptyset$ one has $[\mathsf{P}(\mathsf{W}(L))]\emptyset = \{\varepsilon\}$, so $L \subsetneq \mathsf{P}(\mathsf{W}(L))$. The equality $L = \mathsf{P}(\mathsf{W}(L))$ holds iff L is *downwards closed*, that is, $L(S') = L(S) \cap (S')^\star$ for all $S' \subseteq S \subseteq_{\mathsf{f}} \mathbb{A}$.

The presheaf version of positive word languages and positive closures is as follows:

▶ **Definition 7.2.** Let $L \subseteq V_{\mathbb{I}}^\star$ be a presheaf language in $\mathbf{Set}^{\mathbb{I}}$.

1. The language L is *positive* if $L = KE$ for some (unique) language $K \subseteq V_{\mathbb{F}}^\star$ in $\mathbf{Set}^{\mathbb{F}}$.
2. A *positive closure* of L is a language $\bar{L}$ in $\mathbf{Set}^{\mathbb{F}}$ such that $L \subseteq \bar{L}E$ and $\bar{L}$ is minimal with that property, that is, $\bar{L} \subseteq K$ for every language $K \subseteq V_{\mathbb{F}}^\star$ in $\mathbf{Set}^{\mathbb{F}}$ such that $L \subseteq KE$.

A positive closure is clearly unique; its existence is ensured by the next proposition, which is proved using the universal property of left Kan extensions.

▶ **Proposition 7.3.** *The positive closure of $L \subseteq V_{\mathbb{I}}^*$ is given by the image of the morphism*

$$\varphi\colon \mathsf{Lan}_E(L) \xrightarrow{\mathsf{Lan}_E(\iota)} \mathsf{Lan}_E(V_{\mathbb{I}}^*) \cong \coprod_k \mathsf{Lan}_E(V_{\mathbb{I}}^k) \xrightarrow{\coprod_k \mathsf{can}_k} \coprod_k \mathsf{Lan}_E(V_{\mathbb{I}})^k = \coprod_k V_{\mathbb{F}}^k = V_{\mathbb{F}}^*$$

where $\iota\colon L \hookrightarrow V_{\mathbb{I}}^*$ is the inclusion, the isomorphism witnesses preservation of coproducts by the left adjoint Lan_E, and can_k is the canonical map induced by the product projections.

▶ **Remark 7.4.** A presheaf $P \in \mathbf{Set}^{\mathbb{I}}$ is *strong* if $P = I_\star(X)$ for a strong nominal set X. Since $I_\star$ preserves coproducts, (super-finitary) strong presheaves are exactly (finite) coproducts $\coprod_{j\in J} \mathbb{I}(S_j,-)$ of representable presheaves. By Proposition 5.2 and Proposition 6.9, every super-finitary $\mathbf{Set}^{\mathbb{I}}$-automaton is equivalent to one whose presheaf of states is strong. Given such an automaton A with states $Q = \coprod_{j\in J} \mathbb{I}(S_j,-)$, applying the lifted left adjoint $\overline{\mathsf{Lan}_E}$ yields a super-finitary $\mathbf{Set}^{\mathbb{F}}$-automaton $\bar{A}$ with states $\mathsf{Lan}_E(Q) = \coprod_{j\in J} \mathbb{F}(S_j,-)$, using that Lan_E preserves coproducts and representables (see e.g. [22, Ex. X.3.2]). This is the analogue of Construction 2.7 for presheaf automata. Similar to Proposition 2.8, we have

▶ **Proposition 7.5.** *For every super-finitary nondeterministic* $\mathbf{Set}^{\mathbb{I}}$*-automaton A with a strong presheaf of states, the* $\mathbf{Set}^{\mathbb{F}}$*-automaton $\bar{A} = \overline{\mathsf{Lan}_E}(A)$ accepts the language $\overline{L(A)}$.*

While by definition nondeterministic presheaf automata accept presheaf languages, using Remark 7.1 we can also naturally associate a word language semantics to them:

▶ **Definition 7.6.**

1. The word language *accepted* by a nondeterministic $\mathbf{Set}^{\mathscr{C}}$-automaton A is $\mathsf{W}(L(A)) \subseteq \mathbb{A}^\star$, the word language induced by the presheaf language of A.
2. A word language $L \subseteq \mathbb{A}^\star$ is $\mathbf{Set}^{\mathscr{C}}$*-recognizable* if there exists a super-finitary nondeterministic $\mathbf{Set}^{\mathscr{C}}$-automaton accepting it.

This enables a classification of the expressive power of nondeterministic $\mathbf{Set}^{\mathscr{C}}$-automata:

▶ **Theorem 7.7.**

1. *A word language is NOFA-recognizable iff it is* $\mathbf{Set}^{\mathbb{I}}$*-recognizable.*
2. *A word language is positive and NOFA-recognizable iff it is* $\mathbf{Set}^{\mathbb{F}}$*-recognizable.*

For item 1 one shows that the functors $\bar{I}_\star$ and $\bar{I}^\star$ of (6.2) preserve the accepted word languages of automata. For item 2 one uses Proposition 7.5 and the observation that every nondeterministic $\mathbf{Set}^{\mathbb{F}}$-automaton accepts a positive word language.

This shows that the theory of data languages can be based on presheaves rather than nominal sets [6]. In particular, the conceptual difference between the two approaches (viz. existence of least supports) is largely inessential from the perspective of automata theory.

8 Conclusions and Future Work

We have characterized positive data languages recognizable by NOFAs in terms of register automata, logic, and category theory; see Figure 1 for a summary of our contributions. Our results underline the phenomenon that weak classes of data languages tend to have a rich theory and admit many equivalent perspectives, paralleling classical regular languages over finite alphabets. For example, a similar observation has been made for data languages recognizable by orbit-finite nominal monoids [4,8,9].

The logic $\mathrm{MSO}^{\sim,+}$ defines positive data languages, but is more expressive than NOFAs. Identifying a suitable syntactic fragment of $\mathrm{MSO}^{\sim,+}$ that captures precisely the positive NOFA-recognizable languages remains an open problem. The same holds for the decidability of the satisfiability problem for $\mathrm{MSO}^{\sim,+}$, which is known to be undecidable for $\mathrm{MSO}^{\sim}$ [20]. On a related note, it might be interesting to characterize the expressive power of full $\mathrm{MSO}^{\sim,+}$. Specifically, does it capture precisely the $\mathrm{MSO}^{\sim}$-definable positive languages?

Finally, besides register automata, a number of further automata models for data languages have been proposed, most notably pebble automata [25] and data automata [5,7]. In general, these models differ in their expressive power. However, it is conceivable that some or all of them may become equivalent when restricted to positive data languages.

References

1 Jiří Adámek, Stefan Milius, Lurdes Sousa, and Thorsten Wißmann. On finitary functors. *Theory Appl. Categ.*, 34(37):1134–1164, 2019.

2 Jiří Adámek and Jiří Rosický. *Locally Presentable and Accessible Categories*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1994.

3 Michał Bielecki, Jan Hidders, Jan Paredaens, Jerzy Tyszkiewicz, and Jan Van den Bussche. Navigating with a browser. In *Proc. 29th International Colloquium on Automata, Languages and Programming (ICALP 2002)*, volume 2380 of *Lect. Notes Comput. Sci.*, pages 764–775. Springer, 2002.

4 Mikołaj Bojańczyk. Nominal monoids. *Theory Comput. Syst.*, 53(2):194–222, 2013. `doi:10.1007/s00224-013-9464-1`.

5 Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data trees and XML reasoning. In *Proc. 25th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2006)*, pages 10–19. ACM, 2006.

6 Mikołaj Bojańczyk, Bartek Klin, and Sławomir Lasota. Automata theory in nominal sets. *Log. Methods Comput. Sci.*, 10(3), 2014. `doi:10.2168/LMCS-10(3:4)2014`.

7 Mikołaj Bojańczyk, Anca Muscholl, Thomas Schwentick, Luc Segoufin, and Claire David. Two-variable logic on words with data. In *Proc. 21th IEEE Symposium on Logic in Computer Science (LICS 2006)*, pages 7–16. IEEE Computer Society, 2006.

8 Mikołaj Bojańczyk and Rafał Stefański. Single-use automata and transducers for infinite alphabets. In *Proc. 47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *LIPIcs*, pages 113:1–113:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.ICALP.2020.113`.

9 Thomas Colcombet, Clemens Ley, and Gabriele Puppis. Logics with rigidly guarded data tests. *Log. Methods Comput. Sci.*, 11(3), 2015.

10 Marcelo P. Fiore, Gordon D. Plotkin, and Daniele Turi. Abstract syntax and variable binding. In *Proc. 14th Annual IEEE Symposium on Logic in Computer Science (LICS 1999)*, pages 193–202. IEEE Computer Society, 1999.

11 Murdoch J. Gabbay. Nominal renaming sets (technical report), 2007. URL: `http://gabbay.org.uk/papers/nomrs-tr.pdf`.

12 Murdoch J. Gabbay and Martin Hofmann. Nominal renaming sets. In *Proc. 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2008)*, pages 158–173. Springer, 2008.

13 Murdoch J. Gabbay and Andrew M. Pitts. A new approach to abstract syntax involving binders. In *Proc. 14th Annual IEEE Symposium on Logic in Computer Science (LICS 1999)*, pages 214–224. IEEE Computer Society, 1999.

14 Fabio Gadducci, Marino Miculan, and Ugo Montanari. About permutation algebras, (pre)sheaves and named sets. *High. Order Symb. Comput.*, 19(2-3):283–304, 2006.

15 Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. Generic trace semantics via coinduction. *Log. Methods Comput. Sci.*, 3(4:11):1–36, 2007.

16 Peter T. Johnstone. *Sketches of an Elephant: A Topos Theory Compendium*. Oxford Logic Guides. Oxford Univ. Press, 2002.

17 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994. `doi:10.1016/0304-3975(94)90242-9`.

18 Michael Kaminski and Tony Tan. Regular expressions for languages over infinite alphabets. *Fundam. Informaticae*, 69(3):301–318, 2006.

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

20 Bartek Klin, Sławomir Lasota, and Szymon Torunczyk. Nondeterministic and co-nondeterministic implies deterministic, for data languages. In *Proc. 24th International Conference on Foundations of Software Science and Computation Structures (FOSSACS 2021)*, volume 12650 of *Lect. Notes Comput. Sci.*, pages 365–384. Springer, 2021.

21 Klaas Kürtz, Ralf Küsters, and Thomas Wilke. Selecting theories and nonce generation for recursive protocols. In *Proc. 2007 ACM Workshop on Formal Methods in Security Engineering (FMSE 2007)*, pages 61–70. ACM, 2007.

22 Saunders Mac Lane. *Categories for the Working Mathematician.* Springer, 1971.

23 Stefan Milius and Henning Urbat. Equational axiomatization of algebras with structure. In *Proc. 22nd International Conference on Foundations of Software Science and Computation Structures (FOSSACS 2019)*, volume 11425 of *Lect. Notes Comput. Sci.*, pages 400–417. Springer, 2019. `doi:10.1007/978-3-030-17127-8_23`.

24 Joshua Moerman and Jurriaan Rot. Separation and Renaming in Nominal Sets. In *Proc. 28th EACSL Annual Conference on Computer Science Logic (CSL 2020)*, volume 152 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020.

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

26 Daniela Petrişan. *Investigations into Algebra and Topology over Nominal Sets.* PhD thesis, University of Leicester, 2012.

27 Andrew M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*, volume 57 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, 2013.

28 Jan J. M. M. Rutten. Universal coalgebra: A theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000.

29 Lutz Schröder, Dexter Kozen, Stefan Milius, and Thorsten Wißmann. Nominal automata with name binding. In *Proc. 20th International Conference on Foundations of Software Science and Computation Structures, (FOSSACS 2017)*, volume 10203 of *Lect. Notes Comput. Sci.*, pages 124–142, 2017.

30 Alexandra Silva, Filippo Bonchi, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Generalizing determinization from automata to coalgebras. *Log. Methods Comput. Sci.*, 9(1:9), 2013.

31 Ian Stark. Categorical models for local names. *LISP Symb. Comput.*, 9(1):77–107, 1996.

32 A. Tal. Decidability of inclusion for unification based automata. Master's thesis, Department of Computer Science, Technion – Israel Institute of Technology, 1999.