Parikh images of register automata*

Sławomir Lasota
University of Warsaw
Mohnish Pattathurajan
University of Warsaw

Abstract

As it has been recently shown, Parikh images of languages of nondeterministic one-register automata are rational (but not semilinear in general), but it is still open if the property extends to all register automata. We identify a subclass of nondeterministic register automata, called *hierarchical register automata* (HRA), with the following two properties: every rational language is recognised by a HRA; and Parikh image of the language of every HRA is rational. In consequence, these two properties make HRA an automata-theoretic characterisation of languages of nondeterministic register automata with rational Parikh images.

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

Keywords and phrases Sets with atoms, register automata, Parikh images, rational sets, hierarchical register automata.

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2021.27

Acknowledgements We are grateful to Piotrek Hofman and Arka Ghosh for fruitful discussions.

1 Introduction

Register automata, also know as finite-memory automata, introduced over 25 years ago by Francez and Kaminski [14], are nondeterministic finite-state devices recognising languages over infinite alphabets. They are equipped with a finite number of registers that can store data values (atoms) from an infinite data domain. A register automaton inputs a string of data values (a data word) and compares each consecutive input to its registers; based on this comparison and on the current control state, it chooses a next control state and possibly stores the input value in one of its registers. The only allowed comparisons of data values considered in this paper are equality and inequaltiy tests. An automaton can also guess a fresh data value different from the ones seen currently in the input or stored in registers, and store it in some register (we thus consider nondeterministic register automata *with guessing* [24]). Likewise one may define register context-free grammars [6], [1, Sect.5].

Register automata lack most of the good properties of finite automata, like determinisation or closure properties. In particular, no satisfactory characterisation in terms of rational (regular) expressions is known. Indeed, all known generalisations of Kleene's theorem for register automata either apply to a restricted subclass of the model [17], or introduce an involved syntax significantly extending the concept of rational expressions [19, 18], or rely on a richer algebraic structure than the free monoid of data words [3].

Register automata are expressively equivalent to *orbit-finite automata* [5, 6], a natural extension of finite automata where input alphabets and state spaces are possibly infinite, but finite up to permutation of the data domain (such sets are called *orbit-finite*). Along these lines, we focus on a natural extension of rational expressions, which differ from the classical ones just by allowing for *orbit-finite unions* instead of only finite ones. In other words, we

* This work was partially supported by NCN grant 2019/35/B/ST6/02322 and by the ERC grant INFSYS, agreement no. 950398.

41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2021).
Editors: Mikołaj Bojańczyk and Chandra Chekuri; Article No. 27; pp. 27:1–27:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

consider the class of *rational languages*, defined as the smallest class of languages containing all single-word languages, and closed under concatenation, star, and orbit-finite unions. In particular, the class contains the empty language, all finite and all orbit-finite languages.

Languages of register automata are not rational in general, even in case of deterministic one-register automata. Kleene theorem may be however recovered, at least in case of automata with one register, when commutative images (Parikh images) are considered: the language of every one-register automaton is Parikh-equivalent to (i.e., has the same Parikh image as) a rational language [16]. An analogous result holds for one-register context-free grammars [16].

▶ **Example 1.** Fix the data domain $\textsc{Atoms} = \{0, 1, 2, \ldots\}$. As a working example we will use the language L_1 consisting of all nonempty words over $\textsc{Atoms}$ of length divisible by 3, where every three consecutive letters are pairwise different (we write $\neq(a, b, c)$ as a shorthand for $a \neq b \neq c \neq a$, to concisely express pairwise inequality of three atoms):

$$L_1 \;=\; \{a_1 a_2 \ldots a_{3n} \in \textsc{Atoms}^* \,:\, n \geq 0,\ \neq(a_1, a_2, a_3),\ \neq(a_2, a_3, a_4),\ \neq(a_3, a_4, a_5),\ \ldots\}.$$

The language is recognised by a deterministic two-register automaton but it is not rational (cf. Section 3). It is however Parikh-equivalent to a larger language L_2, where the pairwise inequality constraint is imposed at consecutive disjoint triples of positions only:

$$L_2 \;=\; \{a_1 a_2 \ldots a_{3n} \in \textsc{Atoms}^* \,:\, n \geq 0,\ \neq(a_1, a_2, a_3),\ \neq(a_4, a_5, a_6),\ \ldots\},$$

which is defined by the following rational (regular) expression

$$L_2 \;=\; \Big(\bigcup_{a,b,c \in \textsc{Atoms},\ \neq(a,b,c)} abc \Big)^* \tag{1}$$

and is thus rational. The formal definition of rational languages will be given in Section 3; here we note that the union is indexed by the set $\textsc{Atoms}^{(3)} = \{\langle abc \rangle \in \textsc{Atoms}^3 \,:\, \neq(a, b, c)\}$ of all triples of pairwise-distinct atoms, which is infinite but orbit-finite, i.e., finite up to permutations of $\textsc{Atoms}$ (in fact, it is one orbit).

The language L_1 is Parikh-equivalent to L_2 as every $w = a_1 a_2 \ldots a_{3n} \in L_2$ can be transformed, by swapping letters, to a word in $w' \in L_1$.

Indeed, consider the first two triples (a_1, a_2, a_3) and (a_4, a_5, a_6) in w. We keep the first triple in w'. For the fourth position of w', we choose a letter from $\{a_4, a_5, a_6\} - \{a_2, a_3\}$, say a_6. For the fifth position we choose $\{a_4, a_5\} - \{a_3\}$, say a_4. We note that both the choices are possible due to pigeon-hole principle. Finally, at the sixth position of w' we place the remaining letter a_5. Then we consider next two triples, (a_5, a_4, a_6) and (a_7, a_8, a_9), and treat them analogously by swapping a_7, a_8 and a_9 accordingly. Continuing in this way we finally arrive at a word in $w' \in L_1$. ◀

Contribution. We contribute to understanding of expressive power of nondeterministic register automata (NRA), by investigating sets of *data vectors* obtainable as commutative images (Parikh images) of their languages. Parikh images of rational languages we call rational as well. Here are our contributions:

(1) We identify a syntactic subclass of NRA, called *hierarchical register automata* (HRA).
(2) We show that every rational language is recognised by a HRA.
(3) We show that Parikh images of HRA languages are rational (as a set of data vectors).
(4) As a corollary, we deduce that an NRA has rational Parikh image if, and only if it is Parikh-equivalent to some HRA (with, possibly, more registers).

These results are a step towards the ultimate (but still unreachable) goal: generalise the main result of [16], namely rationality of Parikh images of nondeterministic 1-register automata, to all NRA. Point (3) is an extension from 1-NRA to all HRA. In consequence of (4), the ultimate goal can be equivalently achieved by proving that every nondeterministic register automaton is Parikh-equivalent to a hierarchical one. Finally, we believe that the subclass of HRA (1) is interesting on its own, as it seems to be equally well-behaved as one-register automata.

Related research. Register automata have been intensively studied with respect to their foundational properties [14, 23, 17, 21]. Following the seminal paper of Francez and Kaminski [14], subsequent extensions of the model allow for comparing data values with respect to some fixed relations such as a total order, or introduce alternation, variations on the allowed form of nondeterminism, etc. The model is well known to satisfy almost no semantic equivalences that hold for classical finite automata, in particular register automata admit no satisfactory characterizations in terms of regular expressions [19, 18] or logic [21, 10]. There just are few positive results: simulation of two-way nondeterministic automata by one-way alternating automata with guessing [1]; Myhill-Nerode-style characterisation of languages of deterministic automata [15, 5, 6]; and the well-behaved class of languages definable by orbit-finite monoids [2], characterised in terms of logic [9] and a syntactic subclass of deterministic register automata [8]. Register automata have been also intensively studied with respect to their applications to XML databases and logics [12, 21, 10, 24].

Other extensions of finite-state machines to infinite alphabets include: abstract reformulation or register automata, known as orbit-finite automata, or nominal automata, or automata over atoms) [5, 6, 1]; symbolic automata [11]; pebble automata [20]; and data automata [4, 7] (the list is illustrative).

2 Orbit-finite sets

Sets with atoms. Our definitions rely on basic notions and results of the theory of *sets with atoms* [1], also known as nominal sets [22]. In this section we recall, following [16], what is necessary for understanding of our arguments. This paper is a part of a uniform abstract approach to register automata in the realm of orbit-finite sets with atoms, developed in [5, 6, 1].

Fix a countably infinite set ATOMS, whose elements we call *atoms*. Informally speaking, a set with atoms is a set that can have atoms, or other sets with atoms, as elements. Formally, we define the universe of sets with atoms by a suitably adapted cumulative hierarchy of sets, by transfinite induction: the only set of *rank* 0 is the empty set; and for a cardinal γ, a set of rank γ may contain, as elements, sets of rank smaller than γ as well as atoms. In particular, nonempty subsets $X \subseteq$ ATOMS have rank 1. Sets containing no atoms, whose elements contain no atoms, and so on, we call *pure* (or *atomless*).

Denote by PERM the group of all permutations of ATOMS. Atom permutations π : ATOMS $\to$ ATOMS act on sets with atoms by consistently renaming all atoms in a given set. Formally, by another transfinite induction we define $\pi(X) = \{\pi(x) : x \in X\}$. Via standard set-theoretic encodings of pairs or finite sequences we obtain, in particular, the pointwise action on pairs $\pi(x, y) = (\pi(x), \pi(y))$, and likewise on finite sequences. For pure sets X, $\pi(X) = X$ for every $\pi \in$ PERM.

We restrict to sets with atoms that only depend on finitely many atoms, in the following sense. A *support* of x is any set $S \subseteq$ ATOMS such that the following implication holds for all $\pi \in$ PERM: if $\pi(s) = s$ for all $s \in S$, then $\pi(x) = x$. An element (or set) x is *finitely supported* if it has some finite support; in this case x has *the least support*, denoted SUPP(x), called *the support* of x (cf. [1, Sect. 6]), [22, Prop. 2.3], [6, Cor. 9.4]). Sets supported by $\emptyset$ we call *equivariant*. For instance, given $a, b \in \text{ATOMS}$, the support of the set

$$L_{ab} \;=\; \{a_1a_2 \ldots a_n \in \text{ATOMS}^* \,:\, n \geq 2,\; a_1 \neq a,\; a_n = b\}$$

is $\{a, b\}$; every pure set is equivariant; the support of a sequence $\langle a_1 \ldots a_n\rangle \in \text{ATOMS}^*$, encoded as a set in a standard way, is the set of atoms $\{a_1, \ldots, a_n\}$ appearing in it; and the support of a function $f : \text{ATOMS} \to \mathbb{N}$ such that $\text{DOM}(f) = \{a \in \text{ATOMS} \,:\, f(a) > 0\}$ is finite, is exactly $\text{DOM}(f)$.

From now on, we shall only consider sets with atoms that are hereditarily finitely supported (called briefly *legal*), i.e., ones that are finitely supported, whose every element is finitely supported, and so on.

Orbit-finite sets. Two (elements of) sets with atoms x, y are *in the same orbit* if $\pi(x) = y$ for some $\pi \in \text{PERM}$. This equivalence relation splits every set with atoms X into equivalence classes, which we call *orbits in* X. A (legal) set is *orbit-finite* if it splits into finitely many orbits. Examples of orbit-finite sets are: ATOMS (1 orbit); $\text{ATOMS} - \{a\}$ for some $a \in \text{ATOMS}$ (1 orbit); ATOMS^2 (2 orbits: diagonal $\{\langle a, b\rangle \,:\, a = b\}$ and non-diagonal $\{\langle a, b\rangle \,:\, a \neq b\}$); ATOMS^3 (5 orbits, corresponding to equality types of triples); $\{1, \ldots, n\} \times \text{ATOMS}$ (n orbits, as $\pi(i) = i$ for every $i \in \mathbb{N}$ and $\pi \in \text{PERM}$, since $\{1, \ldots, n\}$ is pure); the set of non-repeating n-tuples of atoms $\text{ATOMS}^{(n)} = \{a_1 \ldots a_n \in \text{ATOMS}^n \,:\, a_i \neq a_j \text{ for every } 1 \leq i < j \leq n\}$ (1 orbit). On the other hand, the set ATOMS^* is an example of an orbit-infinite set.

A finer equivalence relation is defined using *S-atom permutations*, i.e., permutations that fix a finite set $S \subseteq \text{ATOMS}$. Each orbit splits into finitely many *S-orbits* (cf. [1, Sect. 3.2]). For instance, for every $a \in \text{ATOMS}$, the set ATOMS^2 splits into four $\{a\}$-orbits: $\{\langle a, a\rangle\}$, $\{\langle a, b\rangle \,:\, b \neq a\}$, $\{\langle b, a\rangle \,:\, b \neq a\}$, $\{\langle b, c\rangle \,:\, b, c \neq a\}$.

Given a family $(X_i)_{i \in I}$ of sets indexed by an orbit-finite set I, the union $\bigcup_{i \in I} X_i$ we call *orbit-finite union* of sets X_i. (Formally, not only each set X_i is assumed to be legal, but also the indexing function $i \mapsto X_i$.) As an example, consider $(L_{ab})_{b \in \text{ATOMS}}$. The indexing function $b \mapsto L_{ab}$ is supported by $\{a\}$, and so is the union:

$$\bigcup_{b \in \text{ATOMS}} L_{ab} \;=\; \{a_1a_2 \ldots a_n \in \text{ATOMS}^* \,:\, n \geq 2,\; a_1 \neq a\}.$$

Orbit-finite sets are closed under Cartesian products, subsets, and orbit-finite unions: if each of X_i is orbit-finite, their union $\bigcup_{i \in I} X_i$ is orbit-finite too [1, Sect. 3].

3 Rational sets

In this section we recall the definition of rational sets of data words and data vectors introduced in [16], and state and prove its useful closure properties.

Data words and vectors. By a finite multiset over a set (an alphabet) Σ we mean any function $v : \Sigma \to \mathbb{N}$ such that $v(\alpha) = 0$ for all $\alpha \in \Sigma$ except finitely many. We define the *domain* of v as $\text{DOM}(v) = \{\alpha \in \Sigma \,:\, v(\alpha) > 0\}$, and its *size* as $|v| = \sum_{\alpha \in \text{DOM}(v)} v(\alpha)$ (the same notation is used for the size of a set, and for the length of a word). The *Parikh image* (commutative image) of a word $w \in \Sigma^*$ is the multiset $\text{PAR}(w) : \Sigma \to \mathbb{N}$, where $\text{PAR}(w)(\alpha)$ is the number of appearances of a letter $\alpha \in \Sigma$ in w. For a language $L \subseteq \Sigma^*$, its Parikh image is $\text{PAR}(L) = \{\text{PAR}(w) \,:\, w \in L\}$. Two languages $L, L' \subseteq \Sigma^*$ are *Parikh-equivalent* if they have the same Parikh images: $\text{PAR}(L) = \text{PAR}(L')$. Overloading the notation, we write $|w|$ for the *length* of a word w, and hence $|\text{PAR}(w)| = |w|$. We order multisets pointwise: $v \sqsubseteq v'$

if $v(\alpha) \leq v'(\alpha)$ for all $\alpha \in \Sigma$. The zero (empty) multiset $\mathbf{0}$ satisfies $\mathbf{0}(\alpha) = 0$ for every $\alpha \in \Sigma$. Thus $\mathbf{0} = \textsc{Par}(\varepsilon)$. A singleton $\{\alpha\}$ that maps α to 1 and all other letters to 0, is written as α, omitting brackets $\{\}$. Addition of multisets is pointwise: $(v + v')(\alpha) = v(\alpha) + v'(\alpha)$ for every $\alpha \in \Sigma$; likewise subtraction $v - v'$, for $v' \sqsubseteq v$.

When Σ is an orbit-finite alphabet, words $w \in \Sigma^*$ are traditionally called *data words*, languages $L \subseteq \Sigma^*$ are called *data languages*, and finite multisets $v : \Sigma \to \mathbb{N}$ are called *data vectors*.

Orbit-finite unions. Consider a family of sets $\mathcal{X}$. We say that $\mathcal{X}$ is *closed under orbit-finite unions* if for every family $(X_i)_{i\in I}$ of sets $X_i \in \mathcal{X}$ indexed by an orbit-finite set I, the union $\bigcup_{i\in I} X_i$ belongs to $\mathcal{X}$. We instantiate below this abstract definition to families $\mathcal{X}$ of sets of data words and data vectors.

Rational data languages. We consider data languages over a fixed orbit-finite alphabet Σ. As usual, we define concatenation of two data languages $LL' = \{ww' : w \in L, w' \in L'\}$, and the Kleene star (iteration): $L^* = \{w_1 \dots w_n : n \geq 0, w_1, \dots, w_n \in L\}$. Let *rational data languages* be the smallest class of data languages that contains that contains $\{\varepsilon\}$, all singleton languages $\{\sigma\}$ containing a single one-letter word $\sigma \in \Sigma$, and is closed under concatenation, iteration, and orbit-finite unions. In particular the empty language, all finite languages and all orbit-finite ones are rational. For finite Σ we obtain the classical rational (regular) sets. As expected, without the Kleene star we obtain exactly sets of words of bounded length, or equivalently (cf. [16, Lemma 1]) orbit-finite languages.

When convenient, we may speak of a *rational expression*, by which we mean a formal derivation of a rational language according to the closure rules listed above, in the form of well-founded tree. Concretely, a derivation of $\bigcup_{i\in I} L_i$ is the function mapping every $i \in I$ to a derivation of L_i (a node in a tree whose children are labeled by I), a derivation of LL' is a pair of derivations of L and L' (a binary node), a derivation of L^* is just a derivation of L (a unary node), and a derivation of $\{\varepsilon\}$ or $\{\sigma\}$ is a leaf node.

▶ **Example 2.** Continuing Example 1, the language L_2 is rational, as it can be presented by a rational expression:

$$L_2 \;=\; \Big(\bigcup_{a,b,c\in \textsc{Atoms},\ \neq(a,b,c)} \{a\}\{b\}\{c\} \Big)^*.$$

For readability, in the sequel we omit brackets $\{\}$ when denoting singletons, as in (1). On the other hand, one easily shows that the language L_1 is not rational (e.g., using Proposition 12 from Section 4 and Theorem 13 from Section 5).

Rational sets of data vectors. We consider sets of data vectors over a fixed orbit-finite alphabet Σ. Let addition of two sets X, Y of data vectors be defined by Minkowski sum

$$X + Y = \{x + y : x \in X, y \in Y\},$$

and let the additive star X^* contain all finite sums of elements of X:

$$X^* = \{x_1 + \ldots + x_n : n \geq 0, x_1, \ldots, x_n \in X\}.$$

We define *rational sets* of data vectors as the smallest class of sets of data vectors that contains $\{\mathbf{0}\}$, all singletons $\{\sigma\}$ where σ stands for the 'unit' data vector over Σ that maps σ to 1 and all other letters to 0, and is closed under addition, additive star, and orbit-finite unions. In particular, the empty set, all finite sets and all orbit-finite sets of data vectors are rational.

▶ **Example 3.** Continuing Example 2, the Parikh image of L_1 (and L_2) is rational (for readability we keep omitting brackets $\{\}$):

$$\textsc{Par}(L_1) \;=\; \Big(\bigcup_{a,b,c \in \textsc{Atoms},\ \neq(a,b,c)} a+b+c \Big)^*.$$

▷ Claim 4. (1) Rational sets of data vectors are exactly Parikh images of rational data languages. (2) $\textsc{Par}(L)$ is rational if, and only if, L is Parikh-equivalent to a rational data language.

▶ Remark 5. The classical notion of rational sets in an arbitrary monoid ([13, Chapter VII]) can be generalised along the same lines as above to sets with atoms, by considering orbit-finite unions instead of finite ones. In this paper we stick to monoids of data words and data vectors, over an orbit-finite alphabet.

Closure properties. As tools to be used later, we prove that rationality of a language is preserved by the restriction to a subset of its alphabet, as well as by substitution by rational languages. The same preservation property holds for languages with rational Parikh images.

▶ **Lemma 6.** *If a language $L \subseteq \Sigma^*$ has rational Parikh image (resp. is rational) and $\Gamma \subseteq \Sigma$ then the restriction $L \cap \Gamma^*$ has also rational Parikh image (resp. is rational).*

Proof. Intuitively speaking, it is enough to syntactically remove, in the rational expression defining $\textsc{Par}(L)$, every appearance of a letter $\sigma \in \Sigma - \Gamma$.

Formally, we proceed by induction on a derivation of L. By Claim 4(2) we assume, w.l.o.g., that the language L is rational.

The induction base: when $L = \{\sigma\}$ is a singleton, $\sigma \in \Sigma$, then

$$L \cap \Gamma^* \;=\; \begin{cases} L & \text{if } \sigma \in \Gamma, \\ \emptyset & \text{otherwise,} \end{cases}$$

and in each case $L \cap \Gamma^*$ is rational. The induction step follows immediately as restriction commutes with all the operations involved:

$$(LK)\cap\Gamma^* \;=\; (L\cap\Gamma^*)(K\cap\Gamma^*) \qquad L^*\cap\Gamma^* \;=\; (L\cap\Gamma^*)^* \qquad \big(\bigcup_{i\in I} L_i\big)\cap\Gamma^* \;=\; \big(\bigcup_{i\in I} L_i \cap\Gamma^*\big).$$

◀

Consider a language L over an orbit-finite alphabet Σ and a (legal) family of languages $K = (K_\sigma)_{\sigma\in\Sigma}$ over an alphabet Γ, indexed by Σ. We use the anonymous function notation

$$\sigma \quad \mapsto \quad K_\sigma.$$

The *substitution* $L(K)$ is the language over Γ containing all words obtained from some word $\sigma_1\sigma_2\ldots\sigma_n \in L$, by replacing every letter σ_i by some word from K_{σ_i}:

$$L(K) \;=\; \bigcup_{\sigma_1\sigma_2\ldots\sigma_n \in L} K_{\sigma_1} K_{\sigma_2} \ldots K_{\sigma_n}.$$

▶ **Example 7.** As usual, let $L^+ = L^*L$. Consider the language L_1 from Example 1 and $\Sigma = \Gamma = \textsc{Atoms}$. By the equivariant substitution $K_a = a^+$, or $a \;\mapsto\; a^+$, we obtain the language $L_1(K) \subseteq \textsc{Atoms}^*$ containing words, where each three consecutive maximal constant infixes use three distinct letters (each two consecutive maximal constant infixes use two distinct letters by the very definition), and the total number of these infixes is divisible by 3.

▶ **Lemma 8** ([16], Lemma 5). *If L and all languages K_σ have rational Parikh images (resp. are rational) then the substitution $L(K)$ has also rational Parikh image (resp. is rational).*

4 Register automata

We define the model of nondeterministic register automata, and its syntactic subclass of hierarchical automata.

Nondeterministic register automata (NRA). From now on we mostly consider input alphabets of the form $\Sigma = H \times \textsc{Atoms}$, where H is a finite pure (atomless) set.

Let $k \geq 1$. In the sequel we consistently use variables x_i, x'_i, for $1 \leq i \leq k$, to represent the value of ith register at the start (pre-value) and at the end (post-value) of a transition, respectively. We also consistently use the variable y to represent an input atom. A *non-deterministic k-register automaton* (k-NRA) $\mathcal{A}$ consists of: a finite set H (finite component of the alphabet), a finite set of control locations Q, subsets $I, F \subseteq Q$ of initial resp. accepting locations, and a finite set Δ of transition rules of the form

$$(q(x_1, x_2 \dots x_k), \langle h, y\rangle, \varphi, q'(x'_1, x'_2 \dots x'_k)) \tag{2}$$

where $q, q' \in Q$, $h \in H$, and the transition constraint $\varphi(x_1, x_2 \dots x_k, y, x'_1, x'_2 \dots x'_k)$ is a Boolean combination of equalities involving the variables $x_1, x_2 \dots x_k, y, x'_1, x'_2 \dots x'_k$. The constraint specifies possible relation between the register pre-values $(x_1, x_2 \dots x_k)$, input atom (y), and register post-values $(x'_1, x'_2 \dots x'_k)$ resulting from a transition. If φ entails the equality $x_i = x'_i$, we say that the ith register is *preserved* by the transition rule.

A configuration $\langle q, (a_1 a_2 \dots a_k)\rangle \in Q \times \textsc{Atoms}^{(k)}$ of $\mathcal{A}$, written briefly $q(a_1 a_2 \dots a_k)$, consists of a control location $q \in Q$ and (pairwise distinct[1]) register values $a_i \in \textsc{Atoms}$, for $1 \leq i \leq k$. We note that different registers can not store the same value. For each tuple $\mathbf{r} = a_1 a_2 \dots a_k \in \textsc{Atoms}^{(k)}$, atom $b \in \textsc{Atoms}$, and tuple $\mathbf{r}' = a'_1 a'_2 \dots a'_k \in \textsc{Atoms}^{(k)}$ that satisfy the transition constraint, i.e., $(a_1 a_2 \dots a_k, b, a'_1 a'_2 \dots a'_k) \models \varphi$, a rule (2) induces a transition

$$q(a_1 a_2 \dots a_k) \xrightarrow{\langle h, b\rangle} q'(a'_1 a'_2 \dots a'_k)$$

labeled by $\langle h, b\rangle$ from the configuration $q(a_1 a_2 \dots a_k)$ to the configuration $q'(a'_1 a'_2 \dots a'_k)$. The semantics of k-NRA is defined as in case of classical NFA, with configurations considered as states and $\Sigma = H \times \textsc{Atoms}$ as an alphabet. A *run* of $\mathcal{A}$ over a data word $w = \langle h_1, b_1\rangle\langle h_2, b_2\rangle \dots \langle h_n, b_n\rangle \in \Sigma^*$ is any sequence of configurations $q_0(\mathbf{r}_0), q_1(\mathbf{r}_1), \dots, q_n(\mathbf{r}_n)$, related by transitions labeled by consecutive letters of w:

$$q_0(\mathbf{r}_0) \xrightarrow{\langle h_1, b_1\rangle} q_1(\mathbf{r}_1) \xrightarrow{\langle h_2, b_2\rangle} \dots \xrightarrow{\langle h_n, b_n\rangle} q_n(\mathbf{r}_n), \tag{3}$$

where $q_0(\mathbf{r}_0)$ is an initial configuration (i.e., $q_0 \in I$). A run is *accepting* if the ending configuration $q_n(\mathbf{r}_n)$ is accepting (i.e., $q_n \in F$). A data word w is *accepted* by $\mathcal{A}$ if $\mathcal{A}$ has an accepting run over w.

Let $L_{q(\mathbf{r})\, q'(\mathbf{r}')}(\mathcal{A})$ be the set of data words having an accepting run (3) that starts in $q_0(\mathbf{r}_0) = q(\mathbf{r})$ and ends in $q_n(\mathbf{r}_n) = q'(\mathbf{r}')$. The *language* $L(\mathcal{A})$ recognised by $\mathcal{A}$ is defined as:

$$L(\mathcal{A}) = \bigcup_{q \in I, q' \in F, \mathbf{r}, \mathbf{r}' \in \textsc{Atoms}^{(k)}} L_{q(\mathbf{r})\, q'(\mathbf{r}')}(\mathcal{A}). \tag{4}$$

▶ Remark 9. The above definition allows for *guessing*, i.e., an automaton may nondeterministically choose, and store in its register, an atom not yet seen in the input (cf. [24]). In particular, the initial register values are guessed nondeterministically.

[1] Distinctness of register values is not relevant for expressiveness of register automata.

▶ Remark 10. An alphabet $H \times \text{ATOMS}$ and configurations $Q \times \text{ATOMS}^{(k)}$ are orbit-finite. The model of NRA is a special case of the abstract notion of orbit-finite automata (cf. [1, Sect. 5.2]), where alphabets and state spaces may be arbitrary orbit-finite sets. For alphabet of the form $\Sigma = H \times \text{ATOMS}$, where H is pure and finite, NRA are expressively equivalent to orbit-finite automata [1, Sect. 5.2].

Hierarchical register automata (HRA). We define a syntactical subclass of NRA by restricting transition constraints. The idea is to update registers in a hierarchical manner: if a transition rule does not preserve ith register, pre- and post-values of every larger register (jth register, for $j > i$) are unspecified. Formally, a HRA is a NRA where each transition constraint φ has the following form:

$$\varphi \quad \equiv \quad \psi(x_1, x_2, \ldots, x_i, y, x'_i) \ \wedge \bigwedge_{1 \le j < i} x_j = x'_j, \tag{5}$$

for some $i \in \{1, \ldots, k\}$. The sub-formula ψ describes how the post-value of ith register (x'_i) depends on the relation between the input atom (y) and the pre-values of ith register and smaller ones ($x_1, x_2, \ldots, x_i$). Note that all smaller registers are preserved, and larger ones are not mentioned in φ (and hence their pre- and post-values are unspecified, which means that any pre- and post-values are allowed). Note also that the constraint φ allows for updating ith register (according to the sub-constraint ψ) as well as every larger register (arbitrarily); the former we call *specified* update, and the latter one we call *unspecified* one. The number i we call the *level* of the transition constraint, or of the transition (rule) it appears in. As extreme examples, the following all-registers-preserving constraint

$$\bigwedge_{1 \le j \le k} x_j = x'_j \neq y, \tag{6}$$

as well as the most liberal constraint **true** satisfied by any pre- and post-values of registers and any input atom, both are in the syntactic form (5), at level k and 1, respectively.

Intuitively speaking a HRA, when restricted to transition rules of some fixed level i, resembles a NRA with just one (ith) register, with all larger registers removed, and all smaller registers *frozen* to some fixed values. For $i \le k$ and a tuple of atoms $\mathbf{r} \in \text{ATOMS}^{(i)}$, we may define a refined semantics of a k-HRA $\mathcal{A}$ as the language of words accepted by a run where the values of the first (smallest) i registers are continuously $\mathbf{r}$ and hence never change. We denote the so defined language by $L_{\mathbf{r}}(\mathcal{A})$.

W.l.o.g. we may assume that a HRA is *orbitized*, i.e., its every transition constraint $\varphi(x_1, \ldots, x_i, y, x'_1, \ldots, x'_i)$ at level i defines one orbit (one equality type) in ATOMS^{2i+1}. For instance, the constraint (6) defines one orbit, while **true** does not.

▶ **Example 11.** Let H be a singleton, omitted below; we thus consider ATOMS as an alphabet. The following 2-HRA recognises the language L_2 from Example 1. The control locations are $Q = \{q_1, q_2, q_3\}$, with single initial and accepting one $I = F = \{q_3\}$. The automaton has the following three transition rules:

$$(q_3(x_1, x_2),\ y,\ x_1 = x'_1 \neq y \wedge x_2 = x'_2 \neq y,\ q_2(x'_1, x'_2)),$$
$$(q_2(x_1, x_2),\ y,\ x_1 = x'_1 \wedge x_2 = x'_2 = y,\ q_1(x'_1, x'_2)),$$
$$(q_1(x_1, x_2),\ y,\ x_1 = y,\ q_3(x'_1, x'_2)).$$

the first two at level 2 and the last one at level 1. The post-value x'_2 of the second register is unspecified in the last two rules. Moreover, the post-value x'_1 of the first register is also unspecified in the last rule, and therefore the automaton is not orbitized. It can be easily made orbitized by replacing this last rule with the following ones:

$$(q_1(x_1, x_2),\ y,\ x_1 = y = x'_1,\ q_0(x'_1, x'_2)),$$
$$(q_1(x_1, x_2),\ y,\ x_1 = y \neq x'_1,\ q_0(x'_1, x'_2)).$$

It is not difficult to show that in terms of expressiveness HRA are a strict subclass of NRA:

▶ **Proposition 12.** *The language L_1 from Example 1 is not recognised by any* HRA*.*

Proof. Towards contradiction, suppose L_1 is recognised by a k-HRA $\mathcal{A}$. Consider a word $w = a_1 a_2 \dots a_{k+2} \in \text{Atoms}^*$ of length $k+2$ in which all letters are pairwise different ($a_i \neq a_j$ for $i \neq j$) and an accepting run π of $\mathcal{A}$ over w. Let $\mathbf{r}_i$ be the valuation of registers in π after reading a_i.

We observe that each letter a_i, for $i < k+2$, must be stored in a register in the considered run π: a_i it is the value of some register in $\mathbf{r}_i$. Indeed, suppose contrarily that a_i is not the value of any register in $\mathbf{r}_i$. By replacing this letter in w with a_{i+1} we obtain a word w' where two consecutive letters are equal, and hence $w' \notin L_1$. On the other hand the run π is also an accepting run over w', and hence $w' \in L(\mathcal{A})$ – a contradiction.

Therefore we know that a_i is the value of some ℓ_ith register in $\mathbf{r}_i$, for every $i = 1, \dots, k+1$. Note that this register with value a_i is unique, and that it gets its value either by the specified or unspecified update. We claim that $\ell_i < \ell_{i+1}$ for every $i = 1, \dots, k$. Indeed, suppose $\ell_i \geq \ell_{i+1}$ for some i. The inequality implies that either the value a_i stored in ℓ_ith register is overwritten by the specified update (when $\ell_i = \ell_{i+1}$), or *may* be overwritten by an unspecified one (when $\ell_i > \ell_{i+1}$). By replacing a_i in w with a_{i+2} we obtain a word $w'' \notin L_1$. On the other hand the run π is easily modified into an accepting run over w'' by replacing a_i with a_{i+2} in $\mathbf{r}_i$. In consequence, $w'' \in L(\mathcal{A})$ – a contradiction, similarly as before.

We have thus an increasing sequence $1 \leq \ell_1 < \ell_2 < \dots < \ell_{k+1} \leq k$, thus yielding a contradiction. ◀

As an intermediate corollary of Proposition 12 and Theorem 13 (cf. Section 5) we deduce that L_1 is not rational either.

5 Parikh-equivalence of HRA and rational languages

As our main contribution, we prove that Parikh images of rational languages (rational sets of data vectors) coincide with Parikh images of HRA (cf. Corollary 22). This is split into two parts: on one side we prove that rational data languages are recognised by HRA, and on the other side Parikh images of HRA languages are rational (as sets of data vectors):

▶ **Theorem 13.** *Rational data languages are recognised by* HRA*.*

▶ **Theorem 14.** *Parikh images of* HRA *languages are rational.*

Proof of Theorem 13. We proceed by induction on derivation of a rational language. For convenience we assume, w.l.o.g., that each orbit-finite sum is indexed by a subset of $I \subseteq \text{Atoms}^{(n)}$ of non-repeating n-tuples of atoms, for some $n \in \mathbb{N}$. Indeed, every orbit-finite union can be split into a finite union of single-orbit unions, and every single-orbit set J is the image of an equivariant function f from such a set I (cf. [1, Sect. 3.2]), $J = f(I)$, hence

$$\bigcup_{j \in J} L_j \quad = \quad \bigcup_{i \in I} L_{f(i)} \quad = \quad \bigcup_{i \in I} K_i$$

where $K_i = L_{f(i)}$. Under this simplifying assumption we prove, by induction on derivation of a rational language, the following claim (we say that a tuple $\mathbf{s} \in \text{Atoms}^{(n)}$ supports x if the set of n atoms appearing in $\mathbf{s}$ does so):

▷ Claim 15. For every rational language L over an alphabet of the form $\Sigma = H \times \text{Atoms}$, and every tuple $\mathbf{s}$ supporting its derivation, there is a HRA $\mathcal{A}$ such that $L_{\mathbf{s}}(\mathcal{A}) = L$.

We emphasise that we consider supports of *derivations* of rational languages, defined as well-founded trees (cf. Section 3), instead of supports of languages themselves. Clearly, a tuple supporting a derivation of a language also support the language itself.

The induction base, for $L = \{\varepsilon\}$ or $L = \{\sigma\}$ where $\sigma \in \Sigma$, is straightforward. The induction step splits into three cases.

Case 1: $L = L_1\, L_2$. Let $\mathbf{s}$ be a tuple of atoms supporting the derivation of L, and hence also the derivations of L_1 and L_2. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be the HRA which, due to the induction assumption, recognize $L_{\mathbf{s}}(\mathcal{A}_1) = L_1$ and $L_{\mathbf{s}}(\mathcal{A}_2) = L_2$. Let the automaton $\mathcal{A}$ initially run $\mathcal{A}_1$, and from each accepting location of $\mathcal{A}_1$ nondeterministically choose either to continue inside $\mathcal{A}_1$, or to run $\mathcal{A}_2$. We have $L_{\mathbf{s}}(\mathcal{A}) = L$, as required.

Case 2: $L = K^*$. This case is dealt with similarly to the previous one.

Case 3: $L = \bigcup_{i \in I} L_i$. Let $\mathbf{s}$ be a tuple of atoms supporting the derivation of L, and hence also the set I and the mapping $i \mapsto L_i$. Thus the concatenated tuple $\mathbf{s}i$ supports L_i (recall that i is assumed for convenience to be a tuple of atoms). For an $\mathbf{s}$-orbit J in I, let

$$L_J \;=\; \bigcup_{j \in J} L_j \;\subseteq\; L.$$

Consider an arbitrary $\mathbf{s}$-orbit J in I (each orbit is treated separately). Fix an arbitrary element $i \in J$ and an automaton $\mathcal{B}$ such that, due to the induction assumption, recognizes $L_{\mathbf{s}i}(\mathcal{B}) = L_i$. Therefore, for every $j = \pi(i) \in J$, where π is an $\mathbf{s}$-automorphism, the same automaton $\mathcal{B}$ recognizes $L_{\mathbf{s}j}(\mathcal{B}) = L_j$. Let the automaton $\mathcal{A}_J$ initially guess $i \in J$ and put it into the smallest registers not occupied by $\mathbf{s}$, and then run $\mathcal{B}$. We have $L_{\mathbf{s}}(\mathcal{A}_J) = L_J$. The language L is the union of finitely many languages L_J, and hence L is recognized by a HRA that initially chooses an $\mathbf{s}$-orbit J in I and then runs $\mathcal{A}_J$. ◀

Proof of Theorem 14. We now focus on showing that Parikh images of languages of HRA are rational. The proof proceeds by induction on the number of registers.

Induction base. The induction base, i.e., rationality of Parikh images of 1-HRA languages, follows immediately by the following result of [16]:

▶ **Lemma 16** ([16], Theorem 6). *Parikh images of* 1-*NRA languages are rational.*

Altering paths. Before proceeding to the induction step we recall an immediate corollary of another results of [16] (cf. Lemma 17 below). Given a k-HRA $\mathcal{A} = \langle H, Q, I, F, \Delta \rangle$, we define the language $P_{\mathcal{A}}$ over the alphabet[2] $(Q \times \text{Atoms} \times Q) \,\cup\, (H \times \text{Atoms})$ containing words of the form:

$$\langle q_1, a_1, p_1\rangle\langle h_1, b_1\rangle\langle q_2, a_2, p_2\rangle\langle h_2, b_2\rangle \;\ldots\; \langle q_{n-1}, a_{n-1}, p_{n-1}\rangle\langle h_{n-1}, b_{n-1}\rangle\langle q_n, a_n, p_n\rangle \qquad (7)$$

[2] This is the unique place where we consider reacher alphabets than $H \times \text{Atoms}$, for finite H.

$(n \geq 1)$ such that, for $i = 1, \ldots, n-1$, it holds $a_i \neq a_{i+1}$ and

$$p_i(a_i\mathbf{r}) \xrightarrow{\langle h_i, b_i \rangle} q_{i+1}(a_{i+1}\mathbf{r}') \tag{8}$$

is a transition of $\mathcal{A}$ at level 1 for some tuples $\mathbf{r}, \mathbf{r}' \in \text{Atoms}^{(k-1)}$, and such that $q_1 \in I$ and $p_n \in F$. The atoms a_i and a_{i+1} are here pre- and post-values of the first register, and $\mathbf{r}, \mathbf{r}'$ are pre- and post-values of the remaining $k-1$ registers. Words in P are called *altering paths*. Intutively, a letter $\langle q, a, p \rangle$ represents a run of $\mathcal{A}$ starting from a configuration $q(a\mathbf{r}')$ and ending in $p(a\mathbf{r})$, for some $\mathbf{r}, \mathbf{r}' \in \text{Atoms}^{(k-1)}$, such that the first register contains a and is preserved along the run until the automaton reaches the configuration $p(a\mathbf{r})$, from which the automaton finally updates the first register. Along this run other registers may be updated. As an immediate consequence[3] of [16, Lemma 17] we get:

▶ **Lemma 17.** *The altering path language $P_{\mathcal{A}}$ of a 1-HRA $\mathcal{A}$ has rational Parikh image.*

We observe that the altering path language of a k-HRA $\mathcal{A}$ is the same as the altering path language of a 1-HRA $\mathcal{A}'$ obtained from $\mathcal{A}$ by removing all registers except the first (smallest) one, and all transition rules of level greater than 1. Therefore, as an immediate corollary of Lemma 17 we get:

▷ Claim 18. For every $k \geq 1$, the altering path language $P_{\mathcal{A}}$ of k-HRA $\mathcal{A}$ has rational Parikh image.

Induction step. We now proceed to the induction step. To this aim we fix $k > 1$ and assume that languages of HRA with less than k registers have rational Parikh images. We consider a fixed k-HRA $\mathcal{A} = \langle H, Q, I, F, \Delta \rangle$ and aim at showing that Parikh image of $L(A)$ is rational. W.l.o.g. we assume that $\mathcal{A}$ is orbitized. Let $\Sigma = H \times \text{Atoms}$ denote the input alphabet.

We construct a k-HRA $\mathcal{A}_{qp}$ by removing from $\mathcal{A}$ all transition rules that update (i.e., do not preserve) the first register, and by taking q as the only initial location and p as the only accepting one. Intuitively speaking, the first register is *frozen* in $\mathcal{A}_{qp}$, in the sense that it is never updated and thus keeps its initial value a along the whole run. For $a \in \text{Atoms}$, we denote by

$$L_a(\mathcal{A}_{qp}) \; = \bigcup_{\mathbf{r}, \mathbf{s} \in \text{Atoms}^{(k-1)}} L_{q(a\mathbf{r})\, p(a\mathbf{s})}(\mathcal{A}_{qp}) \;\subseteq\; L(\mathcal{A}_{qp})$$

the subset of $L(\mathcal{A}_{qp})$ consisting of words accepted by $\mathcal{A}_{qp}$ by a run where the value of the first register is (continuously) a. We need to deduce from the induction assumption the following claim:

▷ Claim 19. The languages $L_a(\mathcal{A}_{qp})$ have rational Parikh images.

Before proving the above claim we use it to complete the proof Theorem 14. Consider the language $K = P_{\mathcal{A}}(S)$ obtained by applying the following substitution S to the language $P_{\mathcal{A}}$:

$$\langle q, a, p \rangle \;\mapsto\; L_a(\mathcal{A}_{qp}) \qquad\qquad \langle h, b \rangle \;\mapsto\; \{\langle h, b \rangle\}.$$

In words, triples $\langle q, a, p \rangle$ are replaced by any word accepted by $\mathcal{A}_{qp}$ by a run where the value of the first register is continuously a, while pairs $\langle h, b \rangle$ are preserved.

[3] Altering path languages considered in Lemma 17 in [16] start and end in fixed locations. The language $P_{\mathcal{A}}$ is thus a finite union of these languages.

▷ Claim 20. $L(A) = K$.

We argue that both inclusions hold. The inclusion $L(A) \subseteq K$ is shown by factorising each accepting run of $\mathcal{A}$ by transitions that update the first register, of the form (8), so that each word $w \in L(\mathcal{A})$ factorizes into:

$$w \;=\; w_1 \, \langle h_1, b_1 \rangle \, w_2 \, \langle h_2, b_2 \rangle \, \ldots \, w_{n-1} \, \langle h_{n-1}, b_{n-1} \rangle \, w_n, \tag{9}$$

for $w_i \in L_{a_i}(\mathcal{A}_{q_i p_i})$ for some atom a_i and control locations q_i, p_i, and therefore $w \in K$. For the reverse inclusion $K \subseteq L(A)$ consider a word $w \in K$, necessarily of the form (9), due to an altering path as in (7) and accepting runs π_i of $\mathcal{A}_{q_i p_i}$ over words w_i, where the first register is continuously equal a_i along π_i. By concatenating these runs (considered as sequences of configurations) one gets an accepting run $\pi = \pi_1 \pi_2 \ldots \pi_n$ of $\mathcal{A}$ over the word w, as required. The transitions (8) confirm that π is a run since $\mathcal{A}$ is hierarchical: all these transitions are all at level 1 and may perform (unspecified) updates of all other registers.

Having Claims 18, 19 and 20 one easily completes the proof of Theorem 14. Indeed, Parikh image of $K = P_{\mathcal{A}}(S)$ is rational due to Lemma 8, as Parikh images of $P_{\mathcal{A}}$ and all languages $L_a(\mathcal{A}_{qp})$ are so due to Claim 18 and 19, respectively, and therefore the same holds for $L(A)$, due to Claim 20.

Proof of Claim 19. For every $q, p \in Q$ we define a new $(k-1)$-HRA $\mathcal{A}'_{qp}$ that behaves exactly as $\mathcal{A}_{qp}$ except that the first register is removed. The removal of the register is compensated by an additional bit in the finite component of the alphabet of $\mathcal{A}'_{qp}$ that informs the automaton whether the input atom is equal to the (removed) first register or not.

Formally, the new automaton is $\mathcal{A}'_{qp} = \langle \{=, \neq\} \times H, Q, \{q\}, \{p\}, \Delta' \rangle$, where the transition rules Δ' are defined as follows. Due to the assumption that $\mathcal{A}$ is orbitized (and hence so are all automata $\mathcal{A}_{qp}$), its every transition constraint (5) at level i, say, either entails the equality $y = x_1$, or the inequality $y \neq x_1$. The transition rules Δ' are obtained from the transition rules of $\mathcal{A}_{qp}$ (i.e., from transition rules of $\mathcal{A}$ at level greater than 1) by transforming each transition rule

$$(q(x_1, x_2 \ldots x_k), \langle h, y \rangle, \varphi, q'(x'_1, x'_2 \ldots x'_k))$$

of $\mathcal{A}_{qp}$ to the following one:

$$(q(x_1, x_2 \ldots x_k), \langle (\sim, h), y \rangle, \varphi', q'(x'_1, x'_2 \ldots x'_k))$$

where $\sim \in \{=, \neq\}$ is chosen so that φ entails $y \sim x_1$, and φ' is obtained from φ by removing all (in)equalities referring to the first register.

By induction assumption we know that Parikh image of $\mathcal{A}'_{qp}$ is rational, for every $q, p \in Q$. For $a \in \text{ATOMS}$, consider the following sub-alphabet (that fixes, intuitively, the value of the first register to be a):

$$\Sigma_a \;=\; \{ \langle (=, h), a \rangle \,:\, h \in H \} \;\cup\; \{ \langle (\neq, h), b \rangle \,:\, h \in H,\; b \in \text{ATOMS} - \{a\} \} \;\subseteq\; \Sigma,$$

and define the languages L_{qap} as the restriction of $L(\mathcal{A}'_{qp})$ to the sub-alphabet Σ_a:

$$L_{qap} \;:=\; L(\mathcal{A}'_{qp}) \cap (\Sigma_a)^*.$$

By Lemma 6 we have:

▷ Claim 21. Parikh images of the languages L_{qap} are rational.

Finally, we observe that $L_a(\mathcal{A}_{qp})$ is obtained from L_{qap} by applying the substitution (actually, the projection):

$$\langle(\sim, h), b\rangle \quad\mapsto\quad \{\langle h, b\rangle\}$$

and therefore also has rational Parikh image, as required. This completes the proof of Claim 19, and hence also the proof of Theorem 14. ◀

▶ **Corollary 22.** *Parikh images of* HRA *languages and of rational languages coincide.*

▶ **Corollary 23.** *An* NRA *has rational Parikh image if, and only if, it is Parikh-equivalent to some* HRA*.*

References

1 Mikołaj Bojańczyk. Slightly infinite sets. A draft of a book. URL: `https://www.mimuw.edu.pl/~bojan/paper/atom-book`.

2 Mikołaj Bojańczyk. Data monoids. In *Proc. STACS 2011*, volume 9 of *LIPIcs*, pages 105–116. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011.

3 Mikołaj Bojańczyk. Regular expressions for data words. Personal communication, 2020.

4 Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Log.*, 12(4):27:1–27:26, 2011.

5 Mikołaj Bojańczyk, Bartek Klin, and Slawomir Lasota. Automata with group actions. In *Proc. LICS 2011*, pages 355–364, 2011.

6 Mikołaj Bojańczyk, Bartek Klin, and Slawomir Lasota. Automata theory in nominal sets. *Log. Methods Comput. Sci.*, 10(3), 2014.

7 Mikołaj Bojańczyk and Sławomir Lasota. An extension of data automata that captures XPath. *Log. Methods Comput. Sci.*, 8(1), 2012.

8 Mikołaj Bojańczyk and Rafał Stefański. Single-use automata and transducers for infinite alphabets. In *Proc. ICALP 2020*, volume 168 of *LIPIcs*, pages 113:1–113:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

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

10 Thomas Colcombet and Amaldev Manuel. Generalized data automata and fixpoint logic. In *Proc. FSTTCS 2014*, volume 29 of *LIPIcs*, pages 267–278. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014.

11 Loris D'Antoni and Margus Veanes. Minimization of symbolic automata. In *Proc. POPL '14*, pages 541–554. ACM, 2014.

12 Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3):16:1–16:30, 2009.

13 Samuel Eilenberg. *Automata, languages, and machines. A*. Pure and applied mathematics. Academic Press, 1974. URL: `https://www.worldcat.org/oclc/310535248`.

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

15 Nissim Francez and Michael Kaminski. An algebraic characterization of deterministic regular languages over infinite alphabets. *Theor. Comput. Sci.*, 306(1-3):155–175, 2003.

16 Piotr Hofman, Marta Juzepczuk, Slawomir Lasota, and Mohnish Pattathurajan. Parikh's theorem for infinite alphabets. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE, 2021.

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

18 Alexander Kurz, Tomoyuki Suzuki, and Emilio Tuosto. On nominal regular languages with binders. In Lars Birkedal, editor, *Proc. FOSSACS 2012*, volume 7213 of *Lecture Notes in Computer Science*, pages 255–269. Springer, 2012.

19 Leonid Libkin, Tony Tan, and Domagoj Vrgoc. Regular expressions for data words. *J. Comput. Syst. Sci.*, 81(7):1278–1297, 2015.

20 Tova Milo, Dan Suciu, and Victor Vianu. Typechecking for XML transformers. *J. Comput. Syst. Sci.*, 66(1):66–97, 2003.

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

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

23 Hiroshi Sakamoto and Daisuke Ikeda. Intractability of decision problems for finite-memory automata. *Theor. Comput. Sci.*, 231(2):297–308, 2000.

24 Luc Segoufin. Automata and logics for words and trees over an infinite alphabet. In *Proc. CSL 2006*, volume 4207 of *Lecture Notes in Computer Science*, pages 41–57. Springer, 2006.