Nominal Automata with Name Binding

Lutz Schröder[1(✉)], Dexter Kozen[2], Stefan Milius[1], and Thorsten Wißmann[1]

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany
Lutz.Schroeder@fau.de
[2] Cornell University, Ithaca, New York, USA

Abstract. Nominal sets are a convenient setting for languages over infinite alphabets, i.e. data languages. We introduce an automaton model over nominal sets, *regular nondeterministic nominal automata (RNNA)*, which have a natural coalgebraic definition using abstraction sets to capture transitions that read a fresh letter from the input word. We prove a Kleene theorem for RNNAs w.r.t. a simple expression language that extends *nominal Kleene algebra (NKA)* with unscoped name binding, thus remedying the known failure of the expected Kleene theorem for NKA itself. We analyse RNNAs under two notions of freshness: *global* and *local*. Under global freshness, RNNAs turn out to be equivalent to session automata, and as such have a decidable inclusion problem. Under *local* freshness, RNNAs retain a decidable inclusion problem, and translate into register automata. We thus obtain decidability of inclusion for a reasonably expressive class of nondeterministic register automata, with no bound on the number of registers.

1 Introduction

Data languages are languages over infinite alphabets, regarded as modeling the communication of values from infinite data types such as nonces [23], channel names [17], process identifiers [6], URL's [2], or data values in XML documents (see [27] for a summary). There is a plethora of automata models for data languages [3,16,30], which can be classified along several axes. One line of division is between models that use explicit registers and have a finite-state description (generating infinite configuration spaces) on the one hand, and more abstract models phrased as automata over nominal sets [28] on the other hand. The latter have infinitely many states but are typically required to be *orbit-finite*, i.e. to have only finitely many states up to renaming implicitly stored letters. There are correspondences between the two styles; e.g. Bojańczyk, Klin, and Lasota's *nondeterministic orbit-finite automata (NOFA)* [5] are equivalent to Kaminski and Francez' *register automata (RAs)* [18] (originally called finite memory automata), more precisely to RAs with nondeterministic reassignment [20]. A second distinction concerns notions of freshness: *global freshness* requires that the next letter to be consumed has not been seen before, while *local freshness* postulates only that the next letter is distinct from the (boundedly many) letters currently stored in the registers.

J. Esparza and A.S. Murawski (Eds.): FOSSACS 2017, LNCS 10203, pp. 124–142, 2017.
DOI: 10.1007/978-3-662-54458-7_8

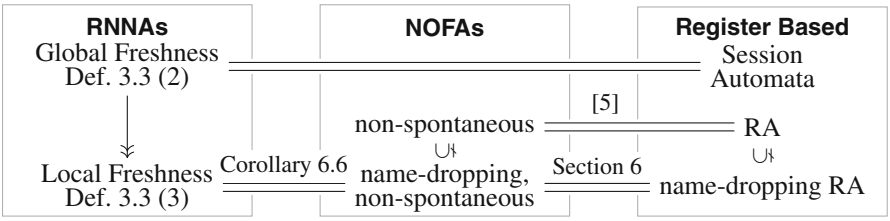


Fig. 1. Expressivity of selected data language formalisms (restricted to empty initial register assignment). FSUBAs are properly contained in name-dropping RA.

Although local freshness looks computationally more natural, nondeterministic automata models (typically more expressive than deterministic ones [21]) featuring local freshness tend to have undecidable inclusion problems. This includes RAs (unless restricted to two registers [18]) and NOFAs [5,27] as well as *variable automata* [16]. *Finite-state unification-based automata (FSUBAs)* [19] have a decidable inclusion problem but do not support freshness. Contrastingly, *session automata*, which give up local freshness in favor of global freshness, have a decidable inclusion problem [6].

Another formalism for global freshness is *nominal Kleene algebra (NKA)* [13]. It has been shown that a slight variant of the original NKA semantics satisfies one half of a Kleene theorem [21], which states that NKA expressions can be converted into a species of nondeterministic nominal automata with explicit *name binding* transitions (the exact definition of these automata being left implicit in op. cit.); the converse direction of the Kleene theorem fails even for deterministic nominal automata.

Here, we introduce *regular bar expressions (RBEs)*, which differ from NKA in making name binding dynamically scoped. RBEs are just regular expressions over an extended alphabet that includes bound letters, and hence are equivalent to the corresponding nondeterministic finite automata, which we call *bar NFAs*. We equip RBEs with two semantics capturing global and local freshness, respectively, with the latter characterized as a quotient of the former: For global freshness, we insist on bound names being instantiated with names not seen before, while in local freshness semantics, we accept also names that have been read previously but will not be used again; this is exactly the usual behaviour of α*-equivalence*, and indeed is formally defined using this notion. Under global freshness, bar NFAs are essentially equivalent to session automata.

We prove bar NFAs to be expressively equivalent to a nondeterministic nominal automaton model with name binding, *regular nondeterministic nominal automata (RNNAs)*. The states of an RNNA form an orbit-finite nominal set; RNNAs are distinguished from NOFAs by having both free and bound transitions and being finitely branching up to α-equivalence of free transitions. This is equivalent to a concise and natural definition of RNNAs as coalgebras for a functor on nominal sets (however, this coalgebraic view is not needed to understand our results). From the equivalence of bar NFAs and RNNAs we obtain

(i) a full Kleene theorem relating RNNAs and RBEs; (ii) a translation of NKA into RBEs, hence, for closed expressions, into session automata; and (iii) decidability in parametrized PSPACE of inclusion for RBEs, implying the known EXPSPACE decidability result for NKA [21].

Under local freshness, RNNAs correspond to a natural subclass of RAs (equivalently, NOFAs) defined by excluding nondeterministic reassignment and by enforcing a policy of *name dropping*, which can be phrased as "at any time, the automaton may nondeterministically lose letters from registers" – thus freeing the register but possibly getting stuck when lost names are expected to be seen later. This policy is compatible with verification problems that relate to scoping, such as 'files that have been opened need to be closed before termination' or 'currently uncommitted transactions must be either committed or explicitly aborted'. Unsurprisingly, RNNAs with local freshness semantics are strictly more expressive than FSUBAs; the relationships of the various models are summarised in Fig. 1. We show that RNNAs nevertheless retain a decidable inclusion problem under local freshness, again in parametrized PSPACE, using an algorithm that we obtain by varying the one for global freshness. This is in spite of the fact that RNNAs (a) do not impose any bound on the number of registers, and (b) allow unrestricted nondeterminism and hence express languages whose complement cannot be accepted by any RA, such as 'some letter occurs twice'.

Further Related Work. A Kleene theorem for *deterministic* nominal automata and expressions with recursion appears straightforward [21]. Kurz et al. [24] introduce regular expressions for languages over words with scoped binding, which differ technically from those used in the semantics of NKA and regular bar expressions in that they are taken only modulo α-equivalence, not the other equations of NKA concerning scope extension of binders. They satisfy a Kleene theorem for automata that incorporate a bound on the nesting depth of binding, rejecting words that exceed this depth.

Data languages are often represented as products of a classical finite alphabet and an infinite alphabet; for simplicity, we use just the set of names as the alphabet. Our unscoped name binders are, under local semantics, similar to the binders in *regular expressions with memory*, which are equivalent to unrestricted register automata [25].

Automata models for data languages, even models beyond register automata such as fresh-register automata [33] and history-register automata [15], often have decidable *emptyness* problems, and their (less expressive) deterministic restrictions then have decidable inclusion problems. Decidability of inclusion can be recovered for nondeterministic or even alternating register-based models by drastically restricting the number of registers, to at most two in the nondeterministic case [18] and at most one in the alternating case [10]. The complexity of the inclusion problem for alternating one-register automata is non-primitive recursive. *Unambiguous register automata* have a decidable inclusion problem and are closed under complement as recently shown by Colcombet et al. [8,9]. RNNAs and unambiguous RAs are incomparable: Closure under complement

implies that the language $L =$'some letter occurs twice' cannot be accepted by an unambiguous RA, as its complement cannot be accepted by any RA [4]. However, L can be accepted by an RNNA (even by an FSUBA). Failure of the reverse inclusion is due to name dropping.

Data walking automata [26] have strong navigational capabilities but no registers, and are incomparable with unrestricted RAs; we do not know how they relate to name-dropping RAs. Their inclusion problem is decidable even under nondeterminism but at least as hard as Petri net reachability, in particular not known to be elementary.

2 Preliminaries

We summarise the basics of *nominal sets*; [28] offers a comprehensive introduction.

Group Actions. Recall that an *action* of a group G on a set X is a map $G \times X \to X$, denoted by juxtaposition or infix $\cdot$, such that $\pi(\rho x) = (\pi\rho)x$ and $1x = x$ for $\pi, \rho \in G$, $x \in X$. A *G-set* is a set X equipped with an action of G. The *orbit* of $x \in X$ is the set $\{\pi x \mid \pi \in G\}$. A function $f : X \to Y$ between G-sets X, Y is *equivariant* if $f(\pi x) = \pi(fx)$ for all $\pi \in G, x \in X$. Given a G-set X, G acts on subsets $A \subseteq X$ by $\pi A = \{\pi x \mid x \in A\}$. For $A \subseteq X$ and $x \in X$, we put

$$\mathsf{fix}x = \{\pi \in G \mid \pi x = x\} \qquad \text{and} \qquad \mathsf{Fix}A = \textstyle\bigcap_{x \in A} \mathsf{fix}x.$$

Note that elements of $\mathsf{fix}\, A$ and $\mathsf{Fix}\, A$ fix A setwise and pointwise, respectively.

Nominal Sets. Fix a countably infinite set $\mathbb{A}$ of *names*, and write G for the group of finite permutations on $\mathbb{A}$. Putting $\pi a = \pi(a)$ makes $\mathbb{A}$ into a G-set. Given a G-set X and $x \in X$, a set $A \subseteq \mathbb{A}$ *supports* x if $\mathsf{Fix}\, A \subseteq \mathsf{fix}\, x$, and x *has finite support* if some finite A supports x. In this case, there is a least set $\mathsf{supp}(x)$ supporting x. We say that $a \in \mathbb{A}$ is *fresh for* x, and write $a \# x$, if $a \notin \mathsf{supp}(x)$. A *nominal set* is a G-set all whose elements have finite support. For every equivariant function f between nominal sets, we have $\mathsf{supp}(fx) \subseteq \mathsf{supp}(x)$. The function supp is equivariant, i.e. $\mathsf{supp}(\pi x) = \pi(\mathsf{supp}(x))$ for $\pi \in G$. Hence $\sharp\mathsf{supp}(x_1) = \sharp\mathsf{supp}(x_2)$ whenever x_1, x_2 are in the same orbit of a nominal set (we use $\sharp$ for cardinality). A subset $S \subseteq X$ is *finitely supported (fs)* if S has finite support with respect to the above-mentioned action of G on subsets; *equivariant* if $\pi x \in S$ for all $\pi \in G$ and $x \in S$ (which implies $\mathsf{supp}(S) = \emptyset$); and *uniformly finitely supported (ufs)* if $\bigcup_{x \in S} \mathsf{supp}(x)$ is finite [32]. We denote by $\mathcal{P}_{\mathsf{fs}}(X)$ and $\mathcal{P}_{\mathsf{ufs}}(X)$ the sets of fs and ufs subsets of a nominal set X, respectively. Any ufs set is fs but not conversely; e.g. the set $\mathbb{A}$ is fs but not ufs. Moreover, any finite subset of X is ufs but not conversely; e.g. the set of words a^n for fixed $a \in \mathbb{A}$ is ufs but not finite. A nominal set X is *orbit-finite* if the action of G on it has only finitely many orbits.

Lemma 2.1 ([12], Theorem 2.29). *If S is ufs, then* $\mathsf{supp}(S) = \bigcup_{x\in S} \mathsf{supp}(x)$.

Lemma 2.2. *Every ufs subset of an orbit-finite set X is finite.*

For a nominal set X we have the *abstraction set* [11]

$$[\mathbb{A}]X = (\mathbb{A} \times X)/{\sim}$$

where $\sim$ abstracts the notion of α-equivalence as known from calculi with name binding, such as the λ-calculus: $(a, x) \sim (b, y)$ iff $(c\,a) \cdot x = (c\,b) \cdot y$ for any fresh c. This captures the situation where x and y differ only in the concrete name given to a bound entity that is called a in x and b in y, respectively. We write $\langle a\rangle x$ for the $\sim$-equivalence class of (a, x). E.g. $\langle a\rangle\{a, d\} = \langle b\rangle\{b, d\}$ in $[\mathbb{A}]\mathcal{P}_\omega(\mathbb{A})$ provided that $d \notin \{a, b\}$.

3 Strings and Languages with Name Binding

As indicated in the introduction, we will take a simplified view of *data languages* as languages over an infinite alphabet; we will use the set $\mathbb{A}$ of names, introduced in Sect. 2, as this alphabet, so that a *data language* is just a subset $A \subseteq \mathbb{A}^*$. Much like nominal Kleene algebra (NKA) [13], our formalism will generate data words from more abstract strings that still include a form of *name binding*. Unlike in NKA, our binders will have unlimited scope to the right, a difference that is in fact immaterial at the level of strings but will be crucial at the level of regular expressions. We write a bound occurrence of $a \in \mathbb{A}$ as $|a$, and define an extended alphabet $\bar{\mathbb{A}}$ by

$$\bar{\mathbb{A}} = \mathbb{A} \cup \{\, |a \mid a \in \mathbb{A}\}.$$

Definition 3.1. A *bar string* is a word over $\bar{\mathbb{A}}$, i.e. an element of $\bar{\mathbb{A}}^*$. The set $\bar{\mathbb{A}}^*$ is made into a nominal set by the letter-wise action of G. The *free names* occurring in a bar string w are those names a that occur in w to the left of any occurrence of $|a$. A bar string is *clean* if its bound letters $|a$ are mutually distinct and distinct from all its free names. We write $\mathsf{FN}(w)$ for the set of free names of w, and say that w is *closed* if $\mathsf{FN}(w) = \emptyset$; otherwise, w is *open*. We define *α-equivalence* $\equiv_\alpha$ on bar strings as the equivalence (*not*: congruence) generated by $w|av \equiv_\alpha w|bu$ if $\langle a\rangle v = \langle b\rangle u$ in $[\mathbb{A}]\bar{\mathbb{A}}^*$ (Sect. 2). We write $[w]_\alpha$ for the α-equivalence class of w. For a bar string w, we denote by $\mathsf{ub}(w) \in \mathbb{A}^*$ (for *unbind*) the word arising from w by replacing all bound names $|a$ with the corresponding free name a.

The set $\mathsf{FN}(w)$ is invariant under α-equivalence, so we have a well-defined notion of *free names* of bar strings modulo $\equiv_\alpha$. Every bar string is α-equivalent to a clean one.

Example 3.2. We have $[ab|cab]_\alpha \neq [ab|aab]_\alpha = [ab|ccb]_\alpha \neq [ap|ccp]_\alpha$ where $\mathsf{FN}(ab|cab) = \mathsf{FN}(ab|aab) = \{a, b\}$. The bar string $ab|aab$ is not clean, and an α-equivalent clean one is $ab|ccb$.

Definition 3.3. A *literal language* is a set of bar strings, and a *bar language* is an fs set of bar strings modulo α-equivalence, i.e. an fs subset of

$$\bar{M} := \bar{\mathbb{A}}^*/\equiv_\alpha. \tag{1}$$

A literal or bar language is *closed* if all bar strings it contains are closed.

Bar languages capture *global freshness*; in fact, the operator N defined by

$$N(L) = \{\mathsf{ub}(w) \mid w \text{ clean}, [w]_\alpha \in L\} \subseteq \mathbb{A}^* \tag{2}$$

is injective on closed bar languages. Additionally, we define the *local freshness semantics* $D(L)$ of a bar language L by

$$D(L) = \{\mathsf{ub}(w) \mid [w]_\alpha \in L\} \subseteq \mathbb{A}^*. \tag{3}$$

That is, $D(L)$ is obtained by taking *all* representatives of α-equivalence classes in L and then removing bars, while N takes only clean representatives. Intuitively, D enforces local freshness by blocking α-renamings of bound names into names that have free occurrences later in the bar string. The operator D fails to be injective; e.g. (omitting notation for α-equivalence classes) $D(\{|a|b, |aa\}) = \mathbb{A}^2 = D(\{|a|b\})$. This is what we mean by our slogan that *local freshness is a quotient of global freshness*.

Remark 3.4. Again omitting α-equivalence classes, we have $D(\{|a|b\}) = \mathbb{A}^2$ because $|a|b \equiv_\alpha |a|a$. On the other hand, $D(\{|a|ba\}) = \{cdc \in \mathbb{A}^3 \mid c \neq d\}$ because $|a|ba \not\equiv_\alpha |a|aa$. We see here that since our local freshness semantics is based on α-equivalence, we can only insist on a letter d being distinct from a previously seen letter c if c will be seen again later. This resembles the process of *register allocation* in a compiler, where program variables are mapped to CPU registers (see [1, Sect. 9.7] for details): Each time the register allocation algorithm needs a register for a variable name ($|v$), any register may be (re)used whose current content is not going to be accessed later.

Remark 3.5. In *dynamic sequences* [14], there are two dynamically scoped constructs $\langle a$ and $a\rangle$ for dynamic allocation and deallocation, respectively, of a name a; in this notation, our $|a$ corresponds to $\langle aa$.

4 Regular Bar Expressions

Probably the most obvious formalism for bar languages are regular expressions, equivalently finite automata, over the extended alphabet $\bar{\mathbb{A}}$. Explicitly:

Definition 4.1. A *nondeterministic finite bar automaton*, or *bar NFA* for short, over $\mathbb{A}$ is an NFA A over $\bar{\mathbb{A}}$. We call transitions of type $q \xrightarrow{a} q$ in A *free transitions* and transitions of type $q \xrightarrow{|a} q$ *bound transitions*. The *literal language* $L_0(A)$ of A

is the language accepted by A as an NFA over $\bar{\mathbb{A}}$. The *bar language* $L_\alpha(A) \subseteq \bar{M}$ (see (1)) accepted by A is defined as

$$L_\alpha(A) = L_0(A)/{\equiv_\alpha}.$$

Generally, we denote by $L_0(q)$ the $\bar{\mathbb{A}}$-language accepted by the state q in A and by $L_\alpha(q)$ the quotient of $L_0(q)$ by α-equivalence. The *degree* $\mathsf{deg}(A)$ of A is the number of names $a \in \mathbb{A}$ that occur in transitions $q \xrightarrow{a} q'$ or $q \xrightarrow{|a} q'$ in A.

Similarly, a *regular bar expression* is a regular expression r over $\bar{\mathbb{A}}$; the *literal language* $L_0(r) \subseteq \bar{\mathbb{A}}^*$ *defined by* r is the language expressed by r as a regular expression, and the *bar language defined by* r is $L_\alpha(r) = L_0(r)/{\equiv_\alpha}$. The *degree* $\mathsf{deg}(r)$ *of* r is the number of names a occurring as either $|a$ or a in r.

Example 4.2. We have $L_\alpha(ac + |cd) = \{ac\} \cup [|cd]_\alpha$. Under local freshness semantics, this bar language contains for example ad, bd, and cd but not dd. $D(L_\alpha((a+|a)^*))$ is the same language as $D(L_\alpha(|a^*))$, even though $(a+|a)^*$ and $|a^*$ define different bar languages.

Remark 4.3. Up to the fact that we omit the finite component of the alphabet often considered in data languages, a *session automaton* [6] is essentially a bar NFA (where free names a are denoted as $a^\uparrow$, and bound names $|a$ as $a^\circledast$). It defines an $\mathbb{A}$-language and interprets bound transitions for $|a$ as binding a to some globally fresh name. In the light of the equivalence of global freshness semantics and bar language semantics in the closed case, session automata are thus essentially the same as bar NFAs; the only difference concerns the treatment of open bar strings: While session automata explicitly reject bar strings that fail to be closed (*well-formed* [6]), a bar NFA will happily accept open bar strings. Part of the motivation for this permissiveness is that we now do not need to insist on regular bar expressions to be closed; in particular, regular bar expressions are closed under subexpressions.

Example 4.4. In terms of $\mathbb{A}$-languages, bar NFAs under global freshness semantics, like session automata, can express the language "all letters are distinct" (as $|a^*$) but not the universal language $\mathbb{A}^*$ [6].

Example 4.5. The bar language $L = \{\epsilon, |ba, |ba|ab, |ba|ab|ba, |ba|ab|ba|ab \dots\}$ (omitting equivalence classes) is defined by the regular bar expression $(|ba|ab)^*(1+|ba)$ and accepted by the bar NFA A with four states s, t, u, v, where s is initial and s and u are final, and transitions $s \xrightarrow{|b} t \xrightarrow{a} u \xrightarrow{|a} v \xrightarrow{b} s$. Under global freshness, the closed bar language $|aL$ defines the language of odd-length words over $\mathbb{A}$ with identical letters in positions 0 and 2 (if any), and with every letter in an odd position being globally fresh and repeated three positions later. Under local freshness, $|aL$ defines the $\mathbb{A}$-language consisting of all odd-length words over $\mathbb{A}$ that contain the same letters in positions 0 and 2 (if any) and repeat every letter in an odd position three positions later (if any) *but no earlier*; that is, the bound names are indeed interpreted as being locally fresh. The reason for this is that, e.g., in the bar string $|a|ba|ab$, α-renaming of the bound

name $|b$ into $|a$ is blocked by the occurrence of a after $|b$; similarly, the second occurrence of $|a$ cannot be renamed into $|b$.

Example 4.6. The choice of fresh letters may restrict the branching later: The language $D(L_\alpha(|a(c + dd))) = \{ac, dc, add, cdd \mid a \in \mathbb{A}\backslash\{c, d\}\}$ contains neither bbb nor cc.

We will see in the sequel that bar NFAs and regular bar expressions are expressively equivalent to several other models, specifically

- under both semantics, to a nominal automaton model with name binding that we call *regular nondeterministic nominal automata*;
- under local freshness, to a class of nondeterministic orbit finite automata [5]; and consequently to a class of register automata.

Nominal Kleene Algebra. We recall that expressions r, s of *nominal Kleene algebra (NKA)* [13], briefly *NKA expressions*, are defined by the grammar

$$r, s ::= 0 \mid 1 \mid a \mid r + s \mid rs \mid r^* \mid \nu a.\, r \qquad (a \in \mathbb{A}).$$

Kozen et al. [21,22] give a semantics of NKA in terms of *ν-languages*. These are fs languages over words with binding, so called *ν-strings*, which are either 1 or ν-regular expressions formed using only names $a \in \mathbb{A}$, sequential composition, and name binding ν, taken modulo the equational laws of NKA [13], including α-equivalence and laws for scope extension of binding. In this semantics, a binder νa is just interpreted as itself, and all other clauses are standard. It is easy to see that the nominal set of ν-strings modulo the NKA laws is isomorphic to the universal bar language $\bar{M}$; one converts bar strings into ν-strings by replacing any occurrence of $|a$ with $\nu a.a$, with the scope of the binder extending to the end of the string. On closed expressions, ν-language semantics is equivalent to the semantics originally defined by Gabbay and Ciancia [13,22], which is given by the operator N defined in (2) (now applied also to languages containing open bar strings). Summing up, we can see NKA as another formalism for bar languages. We will see in the next section that regular bar expressions are strictly more expressive than NKA; the crucial difference is that the name binding construct νa of NKA has a static scope, while bound names $|a$ in regular bar expressions have dynamic scope.

Remark 4.7. On open expressions, the semantics of [13] and [21,22] differ as N may interpret bound names with free names appearing elsewhere in the expression; e.g. the NKA expressions $a + \nu a.\, a$ and $\nu a.\, a$ have distinct bar language semantics $\{a, |a\}$ and $\{|a\}$, respectively, which are both mapped to $\mathbb{A}$ under N. For purposes of expressivity comparisons, we will generally restrict to closed expressions as well as "closed" automata and languages in the sequel. For automata, this typically amounts to the initial register assignment being empty, and for languages to being equivariant subsets of $\bar{\mathbb{A}}^*$.

5 Regular Nondeterministic Nominal Automata

We proceed to develop a nominal automaton model that essentially introduces a notion of configuration space into the picture, and will turn out to be equivalent to bar NFAs. The deterministic restriction of our model has been considered in the context of NKA [21].

Definition 5.1. A *regular nondeterministic nominal automaton (RNNA)* is a tuple $A = (Q, \to, s, F)$ consisting of

- an orbit-finite set Q of states, with an *initial* state $s \in Q$;
- an equivariant subset $\to$ of $Q \times \bar{\mathbb{A}} \times Q$, the *transition relation*, where we write $q \xrightarrow{\alpha} q'$ for $(q, \alpha, q') \in \to$; transitions of type $q \xrightarrow{a} q'$ are called *free*, and those of type $q \xrightarrow{\restriction a} q'$ *bound*;
- an equivariant subset $F \subseteq Q$ of *final* states

such that the following conditions are satisfied:

- The relation $\to$ is *α-invariant*, i.e. closed under α-equivalence of transitions, where transitions $q \xrightarrow{\restriction a} q'$ and $p \xrightarrow{\restriction b} p'$ are *α-equivalent* if $q = p$ and $\langle a \rangle q' = \langle b \rangle p'$.
- The relation $\to$ is *finitely branching up to α-equivalence*, i.e. for each state q the sets $\{(a, q') \mid q \xrightarrow{a} q'\}$ and $\{\langle a \rangle q' \mid q \xrightarrow{\restriction a} q'\}$ are finite (equivalently ufs, by Lemma 2.2).

The *degree* $\mathsf{deg}(A) = \max\{\sharp\mathsf{supp}(q) \mid q \in Q\}$ of A is the maximum size of supports of states in A.

Remark 5.2. For readers familiar with universal coalgebra [29], we note that RNNAs have a much more compact definition in coalgebraic terms, and in fact we regard the coalgebraic definition as evidence that RNNAs are a natural class of automata; however, no familiarity with coalgebras is required to understand the results of this paper. Coalgebraically, an RNNA is simply an orbit-finite coalgebra $\gamma : Q \to FQ$ for the functor F on Nom given by

$$FX = 2 \times \mathcal{P}_{\mathsf{ufs}}(\mathbb{A} \times X) \times \mathcal{P}_{\mathsf{ufs}}([\mathbb{A}]X),$$

together with an initial state $s \in Q$. The functor F is a nondeterministic variant of the functor $KX = 2 \times X^{\mathbb{A}} \times [\mathbb{A}]X$ whose coalgebras are *deterministic nominal automata* [21]. Indeed Kozen et al. [21] show that the ν-languages, equivalently the bar languages, form the final K-coalgebra.

We proceed to define the language semantics of RNNAs.

Definition 5.3. An RNNA A, with data as above, *(literally) accepts* a bar string $w \in \bar{\mathbb{A}}^*$ if $s \xrightarrow{w} q$ for some $q \in F$, where we extend the transition notation $\xrightarrow{w}$ to bar strings in the usual way. The *literal language accepted by* A is the set $L_0(A)$ of bar strings accepted by A, and the *bar language accepted by* A is the quotient $L_\alpha(A) = L_0(A)/\equiv_\alpha$.

A key property of RNNAs is that supports of states evolve in the expected way along transitions (cf. [21, Lemma 4.6] for the deterministic case):

Lemma 5.4. *Let A be an RNNA. Then the following hold.*

1. *If $q \xrightarrow{a} q'$ in A then $\mathsf{supp}(q') \cup \{a\} \subseteq \mathsf{supp}(q)$.*
2. *If $q \xrightarrow{|a} q'$ in A then $\mathsf{supp}(q') \subseteq \mathsf{supp}(q) \cup \{a\}$.*

In fact, the properties in the lemma are clearly also sufficient for ufs branching. From Lemma 5.4, an easy induction shows that for any state q in an RNNA and any w literally accepted by A from q, we have $\mathsf{FN}(w) = \mathsf{supp}([w]_\alpha) \subseteq \mathsf{supp}(q)$. Hence:

Corollary 5.5. *Let A be an RNNA. Then $L_\alpha(A)$ is ufs; specifically, if s is the initial state of A and $w \in L_\alpha(A)$, then $\mathsf{supp}(w) \subseteq \mathsf{supp}(s)$.*

We have an evident notion of α-equivalence of paths in RNNAs, defined analogously as for bar strings. Of course, α-equivalent paths always start in the same state. The set of paths of an RNNA A is closed under α-equivalence. However, this does not in general imply that $L_0(A)$ is closed under α-equivalence; e.g. for A being

$$s() \xrightarrow{|a} t(a) \xrightarrow{|b} u(a,b) \tag{4}$$

(with a, b ranging over distinct names in $\mathbb{A}$), where $s()$ is initial and the states $u(-,-)$ are final, we have $|a|b \in L_0(A)$ but the α-equivalent $|a|a$ is not in $L_0(A)$. Crucially, closure of $L_0(A)$ under α-equivalence is nevertheless without loss of generality, as we show next.

Definition 5.6. An RNNA A is *name-dropping* if for every state q in A and every subset $N \subseteq \mathsf{supp}(q)$ there exists a state $q|_N$ in A that *restricts q to N*; that is, $\mathsf{supp}(q|_N) = N$, $q|_N$ is final if q is final, and $q|_N$ has at least the same incoming transitions as q (i.e. whenever $p \xrightarrow{\alpha} q$ then $p \xrightarrow{\alpha} q|_N$), and as many of the outgoing transitions of q as possible; i.e. $q|_N \xrightarrow{a} q'$ whenever $q \xrightarrow{a} q'$ and $\mathsf{supp}(q') \cup \{a\} \subseteq N$, and $q|_N \xrightarrow{|a} q'$ whenever $q \xrightarrow{|a} q'$ and $\mathsf{supp}(q') \subseteq N \cup \{a\}$.

The counterexample shown in (4) fails to be name-dropping, as no state restricts $q = u(a,b)$ to $N = \{b\}$. The following lemma shows that closure under α-equivalence is restored under name-dropping:

Lemma 5.7. *Let A be a name-dropping RNNA. Then $L_0(A)$ is closed under α-equivalence, i.e. $L_0(A) = \{w \mid [w]_\alpha \in L_\alpha(A)\}$.*

Finally, we can close a given RNNA under name dropping, preserving the bar language:

Lemma 5.8. *Given an RNNA of degree k with n orbits, there exists a bar language-equivalent name-dropping RNNA of degree k with at most $n2^k$ orbits.*

Proof (Sketch). From an RNNA A, construct an equivalent name-dropping RNNA with states of the form

$$q|_N := \mathsf{Fix}(N)q$$

where q is a state in A, $N \subseteq \mathsf{supp}(q)$, and $\mathsf{Fix}(N)q$ denotes the orbit of q under $\mathsf{Fix}(N)$. The final states are the $q|_N$ with q final in A, and the initial state is $s|_{\mathsf{supp}(s)}$, where s is the initial state of A. As transitions, we take

- $q|_N \xrightarrow{a} q'|_{N'}$ whenever $q \xrightarrow{a} q'$, $N' \subseteq N$, and $a \in N$, and
- $q|_N \xrightarrow{!a} q'|_{N'}$ whenever $q \xrightarrow{!b} q''$, $N'' \subseteq \mathsf{supp}(q'') \cap (N \cup \{b\})$, and $\langle a\rangle(q'|_{N'}) = \langle b\rangle(q''|_{N''})$. □

Example 5.9. Closing the RNNA from (4) under name dropping as per Lemma 5.8 yields additional states that we may denote $u(\bot, b)$ (among others), with transitions $t(a) \xrightarrow{!b} u(\bot, b)$; now, $\langle b\rangle u(\bot, b) = \langle a\rangle u(\bot, a)$, so $!a!a$ is (literally) accepted.

Equivalence to Bar NFAs. We proceed to show that RNNAs are expressively equivalent to bar NFAs by providing mutual translations. In consequence, we obtain a Kleene theorem connecting RNNAs and regular bar expressions.

Construction 5.10. We construct an RNNA $\bar{A}$ from a given bar NFA A with set Q of states, already incorporating closure under name dropping as per Lemma 5.8. For $q \in Q$, put $N_q = \mathsf{supp}(L_\alpha(q))$. The set $\bar{Q}$ of states of $\bar{A}$ consists of pairs

$$(q, \pi F_N) \qquad (q \in Q,\ N \subseteq N_q)$$

where F_N abbreviates $\mathsf{Fix}(N)$ and πF_N denotes a left coset. Left cosets for F_N can be identified with injective renamings $N \to \mathbb{A}$; intuitively, $(q, \pi F_N)$ restricts q to N and renames N according to π. (That is, we construct a configuration space, as in other translations into NOFAs [5,7]; here, we create virtual registers according to $\mathsf{supp}(L_\alpha(q))$.) We let G act on states by $\pi_1 \cdot (q, \pi_2 F_N) = (q, \pi_1\pi_2 F_N)$. The initial state of $\bar{A}$ is (s, F_{N_s}), where s is the initial state of A; a state $(q, \pi F_N)$ is final in $\bar{A}$ iff q is final in A. Free transitions in $\bar{A}$ are given by

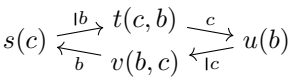
$$(q, \pi F_N) \xrightarrow{\pi(a)} (q', \pi F_{N'}) \text{ whenever } q \xrightarrow{a} q' \text{ and } N' \cup \{a\} \subseteq N$$

and bound transitions by

$$(q, \pi F_N) \xrightarrow{!a} (q', \pi' F_{N'}) \text{ whenever } q \xrightarrow{!b} q', N' \subseteq N \cup \{b\}, \langle a\rangle \pi' F_{N'} = \langle \pi(b)\rangle \pi F_{N'}.$$

Theorem 5.11. *$\bar{A}$ is a name-dropping RNNA with at most $|Q|2^{\mathsf{deg}(A)}$ orbits, $\mathsf{deg}(\bar{A}) = \mathsf{deg}(A)$, and $L_\alpha(\bar{A}) = L_\alpha(A)$.*

Example 5.12. The above construction converts the bar NFA A of Example 4.5, i.e. the expression $(|ba|ab)^*(1 + |ba)$, into an RNNA that is similar to the one appearing in the counterexample to one direction of the Kleene theorem for NKA [21] (cf. Remark 5.15): By the above description of left cosets for F_N, we annotate every state q with a list of $\sharp\mathsf{supp}(L_\alpha(q))$ entries that are either (pairwise distinct) names or $\perp$, indicating that the corresponding name from $\mathsf{supp}(L_\alpha(q))$ has been dropped. We can draw those orbits of the resulting RNNA that have the form $(q, \pi N_q)$, i.e. do not drop any names, as



for $b \neq c$, with $s(c)$, $u(b)$ final for all $b, c \in \mathbb{A}$, and $s(c)$ initial.

Additional states then arise from name dropping; e.g. for t we have additional states $t(\perp, b)$, $t(c, \perp)$, and $t(\perp, \perp)$, all with a $|b$-transition from $s(c)$. The states $t(\perp, \perp)$ and $t(\perp, b)$ have no outgoing transitions, while $t(c, \perp)$ has a c-transition to $u(\perp)$.

We next present the reverse construction, i.e. given an RNNA A we extract a bar NFA A_0 (a subautomaton of A) such that $L_\alpha(A_0) = L_\alpha(A)$.

Put $k = \mathsf{deg}(A)$. We fix a set $\mathbb{A}_0 \subseteq \mathbb{A}$ of size $\sharp\mathbb{A}_0 = k$ such that $\mathsf{supp}(s) \subseteq \mathbb{A}_0$ for the initial state s of A, and a name $* \in \mathbb{A} - \mathbb{A}_0$. The states of A_0 are those states q in A such that $\mathsf{supp}(q) \subseteq \mathbb{A}_0$. As this implies that the set Q_0 of states in A_0 is ufs, Q_0 is finite by Lemma 2.2. For $q, q' \in Q_0$, the free transitions $q \xrightarrow{a} q'$ in A_0 are the same as in A (hence $a \in \mathbb{A}_0$ by Lemma 5.4.1). The bound transitions $q \xrightarrow{|a} q'$ in A_0 are those bound transitions $q \xrightarrow{|a} q'$ in A such that $a \in \mathbb{A}_0 \cup \{*\}$. A state is final in A_0 iff it is final in A. The initial state of A_0 is $s \in Q_0$.

Theorem 5.13. *The number of states in the bar NFA A_0 is linear in the number of orbits of A and exponential in $\mathsf{deg}(A)$. Moreover, $\mathsf{deg}(A_0) \leq \mathsf{deg}(A) + 1$, and $L_\alpha(A_0) = L_\alpha(A)$.*

Combining this with Theorem 5.11, we obtain the announced equivalence result:

Corollary 5.14. *RNNAs are expressively equivalent to bar NFAs, hence to regular bar expressions.*

This amounts to a *Kleene theorem for RNNAs*. The decision procedure for inclusion (Sect. 7) will use the equivalence of bar NFAs and RNNAs, essentially running a bar NFA in synchrony with an RNNA.

Remark 5.15. It has been shown in that an NKA expression r can be translated into a nondeterministic nominal automaton whose states are the so-called *spines* of r, which amounts to one direction of a Kleene theorem [21]. One can show that the spines in fact form an RNNA, so that NKA embeds into regular bar expressions. The automata-to-NKA direction of the Kleene theorem fails

even for deterministic nominal automata, i.e. regular bar expressions are strictly more expressive than NKA. Indeed, the regular bar expression $(|ba|ab)^*(1+|ba)$ of Example 4.5 defines a language that cannot be defined in NKA because it requires unbounded nesting of name binding [21].

6 Name-Dropping Register Automata

We next relate RNNAs to two equivalent models of local freshness, nondeterministic orbit-finite automata [5] and register automata (RAs) [18]. RNNAs necessarily only capture subclasses of these models, since RAs have an undecidable inclusion problem [18]; the distinguishing condition is a version of name-dropping.

Definition 6.1. [5] A nondeterministic orbit-finite automaton (NOFA) A consists of an orbit finite set Q of states, two equivariant subsets $I, E \subseteq Q$ of *initial* and *final* states, respectively, and an equivariant transition relation $\to \subseteq Q \times \mathbb{A} \times Q$, where we write $q \xrightarrow{a} p$ for $(q,a,p) \in \to$. The $\mathbb{A}$-language $L(A) = \{w \mid A \text{ accepts } w\}$ *accepted* by A is defined in the standard way: extend the transition relation to words $w \in \mathbb{A}^*$ as usual, and then say that A *accepts* w if there exist an initial state q and a final state p such that $q \xrightarrow{w} p$. A *DOFA* is a NOFA with a deterministic transition relation.

Remark 6.2. A more succinct equivalent presentation of NOFAs is as orbit-finite coalgebras $\gamma : Q \to GQ$ for the functor

$$GX = 2 \times \mathcal{P}_{\mathsf{fs}}(\mathbb{A} \times X) \qquad (2 = \{\top, \bot\})$$

on the category Nom of nominal sets and equivariant maps, together with an equivariant subset of initial states.

More precisely speaking, NOFAs are equivalent to *RAs with nondeterministic reassignment* [5,20]. RAs are roughly described as having a finite set of registers in which names from the current word can be stored if they are *locally fresh*, i.e. not currently stored in any register; transitions are labeled with register indices k, meaning that the transition accepts the next letter if it equals the content of register k. In the equivalence with NOFAs, the names currently stored in the registers correspond to the support of states.

To enable a comparison of RNNAs with NOFAs over $\mathbb{A}$ (Sect. 5), we restrict our attention in the following discussion to RNNAs that are *closed*, i.e. whose initial state has empty support, and therefore accept equivariant $\mathbb{A}$-languages. We can convert a closed RNNA A into a NOFA $D(A)$ accepting $D(L_\alpha(A))$ by simply replacing every transition $q \xrightarrow{|a} q'$ with a transition $q \xrightarrow{a} q'$. We show that the image of this translation is a natural class of NOFAs:

Definition 6.3. A NOFA A is *non-spontaneous* if $\mathsf{supp}(s) = \emptyset$ for initial states s, and

$$\mathsf{supp}(q') \subseteq \mathsf{supp}(q) \cup \{a\} \quad \text{whenever} \quad q \xrightarrow{a} q'.$$

(In words, A is non-spontaneous if transitions $q \xrightarrow{a} q'$ in A create no new names other than a in q'.) Moreover, A is *α-invariant* if $q \xrightarrow{a} q''$ whenever $q \xrightarrow{b} q'$, $b \# q$, and $\langle a \rangle q'' = \langle b \rangle q'$ (this condition is automatic if $a \# q$). Finally, A is *name-dropping* if for each state q and each set $N \subseteq \mathsf{supp}(q)$ of names, there exists a state $q|_N$ that *restricts q to N*, i.e. $\mathsf{supp}(q|_N) = N$, $q|_N$ is final if q is final, and

- $q|_N$ has at least the same incoming transitions as q;
- whenever $q \xrightarrow{a} q'$, $a \in \mathsf{supp}(q)$, and $\mathsf{supp}(q') \cup \{a\} \subseteq N$, then $q|_N \xrightarrow{a} q'$;
- whenever $q \xrightarrow{a} q'$, $a \# q$, and $\mathsf{supp}(q') \subseteq N \cup \{a\}$, then $q|_N \xrightarrow{a} q'$.

Proposition 6.4. *A NOFA is of the form $D(B)$ for some (name-dropping) RNNA B iff it is (name-dropping and) non-spontaneous and α-invariant.*

Proposition 6.5. *For every non-spontaneous and name-dropping NOFA, there is an equivalent non-spontaneous, name-dropping, and α-invariant NOFA.*

In combination with Lemma 5.7, these facts imply

Corollary 6.6. *Under local freshness semantics, RNNAs are expressively equivalent to non-spontaneous name-dropping NOFAs.*

Corollary 6.7. *The class of languages accepted by RNNAs under local freshness semantics is closed under finite intersections.*

Proof (Sketch). Non-spontaneous name-dropping NOFAs are closed under the standard product construction. □

Remark 6.8. Every DOFA is non-spontaneous. Moreover, RAs are morally non-spontaneous according to their original definition, i.e. they can read names from the current word into the registers but cannot guess names nondeterministically [18,27]; the variant of register automata that is equivalent to NOFAs [5] in fact allows such *nondeterministic reassignment* [20]. This makes unrestricted NOFAs strictly more expressive than non-spontaneous ones [18,34]. Name-dropping restricts expressivity further, as witnessed by the language $\{ab \mid a \neq b\}$ mentioned above. In return, it buys decidability of inclusion (Sect. 7), while for non-spontaneous NOFAs even universality is undecidable [5,27]. DOFAs are incomparable to RNNAs under local freshness semantics—the language "the last letter has been seen before" is defined by the regular bar expression $(\mid b)^* \mid a (\mid b)^* a$ but not accepted by any DOFA.

Name-Dropping Register Automata and FSUBAs. In consequence of Corollary 6.6 and the equivalence between RAs and nonspontaneous NOFAs, we have that RNNAs are expressively equivalent to *name-dropping* RAs, which we just define as those RAs that map to name-dropping NOFAs under the translation given in [5]. We spend a moment on identifying a more concretely defined class of *forgetful* RAs that are easily seen to be name-dropping. We expect that forgetful RAs are as expressive as name-dropping RAs but are currently more

interested in giving a compact description of a class of name-dropping RAs to clarify expressiveness.

We use the very general definition of RAs given in [5]: An RA with n registers consists of a set C of *locations* and for each pair (c, c') of locations a *transition constraint* ϕ. *Register assignments* $w \in R := (\mathbb{A} \cup \{\bot\})^n$ determine the, possibly undefined, contents of the n registers, and *configurations* are elements of $C \times R$. Transition constraints are equivariant subsets $\phi \subseteq R \times \mathbb{A} \times R$, and $(w, a, v) \in \phi$ means that from configuration (c, w) the RA can nondeterministically go to (c', v) under input a. Transition constraints have a syntactic representation in terms of Boolean combinations of certain equations. The NOFA generated by an RA just consists of its configurations.

For $w \in R$ and $N \subseteq \mathbb{A}$ we define $w|_N \in R$ by $(w|_N)_i = w_i$ if $w_i \in N$, and $(w|_N)_i = \bot$ otherwise. An RA is *forgetful* if it generates a non-spontaneous NOFA and for every configuration (c, w) and every N, $(c, w|_N)$ restricts (c, w) to N in the sense of Definition 6.3; this property is equivalent to evident conditions on the individual transition constraints. In particular, it is satisfied if all transition constraints of the RA are conjunctions of the evident non-spontaneity restriction (letters in the poststate come from the input or the prestate) with a positive Boolean combination of the following:

- $\mathsf{cmp}_i = \{(w, a, v) \mid w_i = a\}$ (block unless register i contains the input)
- $\mathsf{store}_i = \{(w, a, v) \mid v_i \in \{\bot, a\}\}$ (store the input in register i or forget)
- $\mathsf{fresh}_i = \{(w, a, v) \mid a \neq w_i\}$ (block if register i contains the input)
- $\mathsf{keep}_{ji} = \{(w, a, v) \mid v_i \in \{\bot, w_j\}\}$ (copy register j to register i, or forget)

FSUBAs [19] can be translated into name-dropping RAs. Unlike FSUBAs, forgetful RAs do allow for freshness constraints. E.g. the language $\{aba \mid a \neq b\}$ is accepted by the forgetful RA $c_0 \xrightarrow{\mathsf{store}_1} c_1 \xrightarrow{\mathsf{fresh}_1 \wedge \mathsf{keep}_{11}} c_2 \xrightarrow{\mathsf{cmp}_1} c_3$, with c_3 final. Note how store and keep will possibly lose the content of register 1 but runs where this happens will not get past cmp_1.

7 Deciding Inclusion under Global and Local Freshness

We next show that under both global and local freshness, the inclusion problem for bar NFAs (equivalently regular bar expressions) is in ExpSpace. For global freshness, this essentially just reproves the known decidability of inclusion for session automata [6] (Remark 4.3; the complexity bound is not stated in [6] but can be extracted), while the result for local freshness appears to be new. Our algorithm differs from [6] in that it exploits name dropping; we describe it explicitly, as we will modify it for local freshness.

Theorem 7.1. *The inclusion problem for bar NFAs is in* ExpSpace*; more precisely, the inclusion $L_\alpha(A_1) \subseteq L_\alpha(A_2)$ can be checked using space polynomial in the size of A_1 and A_2 and exponential in $\mathsf{deg}(A_2)\log(\mathsf{deg}(A_1) + \mathsf{deg}(A_2) + 1)$.*

The theorem can be rephrased as saying that bar language inclusion of NFA is in parametrized polynomial space (para-PSpace) [31], the parameter being the degree.

Proof (Sketch). Let A_1, A_2 be bar NFAs with initial states s_1, s_2. We exhibit an NExpSpace procedure to check that $L_\alpha(A_1)$ is *not* a subset of $L_\alpha(A_2)$, which implies the claimed bound by Savitch's theorem. It maintains a state q of A_1 and a set Ξ of states in the name-dropping RNNA $\bar{A}_2$ generated by A_2 as described in Construction 5.10, with q initialized to s_1 and Ξ to $\{(s_2, \mathsf{id}F_{N_{s_2}})\}$. It then iterates the following:

1. Guess a transition $q \xrightarrow{\alpha} q'$ in A_1 and update q to q'.
2. Compute the set Ξ' of all states of $\bar{A}_2$ reachable from states in Ξ via α-transitions (literally, i.e. not up to α-equivalence) and update Ξ to Ξ'.

The algorithm terminates successfully and reports that $L_\alpha(A_1) \not\subseteq L_\alpha(A_2)$ if it reaches a final state q of A_1 while Ξ contains only non-final states.

Correctness of the algorithm follows from Theorem 5.11 and Lemma 5.7. For space usage, first recall that cosets πF_N can be represented as injective renamings $N \to \mathbb{A}$. Note that Ξ will only ever contain states $(q, \pi F_N)$ such that the image πN of the corresponding injective renaming is contained in the set P of names occurring literally in either A_1 or A_2. In fact, at the beginning, $\mathsf{id}N_{s_2}$ consists only of names literally occurring in A_2, and the only names that are added are those occurring in transitions guessed in Step 7, i.e. occurring literally in A_1. So states $(q, \pi F_N)$ in Ξ can be coded using partial functions $N_q \rightharpoonup P$. Since $\sharp P \leq \mathsf{deg}(A_1) + \mathsf{deg}(A_2)$, there are at most $k \cdot (\mathsf{deg}(A_1) + \mathsf{deg}(A_2) + 1)^{\mathsf{deg}(A_2)} = k \cdot 2^{\mathsf{deg}(A_2)\log(\mathsf{deg}(A_1)+\mathsf{deg}(A_2)+1)}$ such states, where k is the number of states of A_2. □

Remark 7.2. The translation from NKA expressions to bar NFAs (Remark 5.15) increases expression size exponentially but the degree only linearly. Theorem 7.1 thus implies the known ExpSpace upper bound on inclusion for NKA expressions [21].

We now adapt the inclusion algorithm to local freshness semantics. We denote by $\sqsubseteq$ the preorder (in fact: order) on $\bar{\mathbb{A}}^*$ generated by $wav \sqsubseteq w\lfloor a v$.

Lemma 7.3. *Let L_1, L_2 be bar languages accepted by RNNA. Then $D(L_1) \subseteq D(L_2)$ iff for each $[w]_\alpha \in L_1$ there exists $w' \sqsupseteq w$ such that $[w']_\alpha \in L_2$.*

Corollary 7.4. *Inclusion $D(L_\alpha(A_1)) \subseteq D(L_\alpha(A_2))$ of bar NFAs (or regular bar expressions) under local freshness semantics is in para-*PSpace*, with parameter $\mathsf{deg}(A_2)\log(\mathsf{deg}(A_1) + \mathsf{deg}(A_2) + 1)$.*

Proof. By Lemma 7.3, we can use a modification of the above algorithm where Ξ' additionally contains states of $\bar{A}_2$ reachable from states in Ξ via $\lfloor a$-transitions in case α is a free name a. □

8 Conclusions

We have studied the global and local freshness semantics of *regular nondeterministic nominal automata*, which feature explicit name-binding transitions. We have

shown that RNNAs are equivalent to session automata [6] under global freshness and to *non-spontaneous* and *name-dropping* nondeterministic orbit-finite automata (NOFAs) [5] under local freshness. Under both semantics, RNNAs are comparatively well-behaved computationally, and in particular admit inclusion checking in parameterized polynomial space. While this reproves known results on session automata under global freshness, decidability of inclusion under local freshness appears to be new. Via the equivalence between NOFAs and register automata (RAs), we in fact obtain a decidable class of RAs that allows unrestricted non-determinism and any number of registers.

Acknowledgements. We thank Charles Paperman for useful discussions, and the anonymous reviewers of an earlier version of the paper for insightful comments that led us to discover the crucial notion of name dropping. Erwin R. Catesbeiana has commented on the empty bar language.

References

1. Aho, A.V., Sethi, R., Ullman, J.D.: Compilers: Principles, Techniques, and Tools. Addison-Wesley Longman Publishing Co., Inc., Boston (1986)
2. Bielecki, M., Hidders, J., Paredaens, J., Tyszkiewicz, J., Bussche, J.: Navigating with a browser. In: Widmayer, P., Eidenbenz, S., Triguero, F., Morales, R., Conejo, R., Hennessy, M. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 764–775. Springer, Heidelberg (2002). doi:10.1007/3-540-45465-9_65
3. Bojańczyk, M.: Automata for data words and data trees. In: Rewriting Techniques and Applications, RTA 2010. LIPIcs, vol. 6, pp. 1–4. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2010)
4. Bojańczyk, M.: Computation in Sets with Atoms. http://atoms.mimuw.edu.pl/wp-content/uploads/2014/03/main.pdf
5. Bojanczyk, M., Klin, B., Lasota, S.: Automata theory in nominal sets. Log. Methods Comput. Sci. **10**, 1–44 (2014)
6. Bollig, B., Habermehl, P., Leucker, M., Monmege, B.: A robust class of data languages and an application to learning. Log. Meth. Comput. Sci. **10**, 1–23 (2014)
7. Ciancia, V., Tuosto, E.: A novel class of automata for languages on infinite alphabets. Technical report, University of Leicester, cS-09-003 (2009)
8. Colcombet, T., Puppis, G., Skrypczak, M.: Unambiguous register automata, preprint
9. Colcombet, T.: Unambiguity in automata theory. In: Shallit, J., Okhotin, A. (eds.) DCFS 2015. LNCS, vol. 9118, pp. 3–18. Springer, Cham (2015). doi:10.1007/978-3-319-19225-3_1
10. Demri, S., Lazic, R.: LTL with the freeze quantifier and register automata. ACM Trans. Comput. Log. **10**, 16:1–16:30 (2009)
11. Gabbay, M., Pitts, A.: A new approach to abstract syntax involving binders. In: Logic in Computer Science, LICS 1999, pp. 214–224. IEEE Computer Society (1999)
12. Gabbay, M.J.: Foundations of nominal techniques: logic and semantics of variables in abstract syntax. Bull. Symbolic Logic **17**(2), 161–229 (2011)
13. Gabbay, M.J., Ciancia, V.: Freshness and name-restriction in sets of traces with names. In: Hofmann, M. (ed.) FoSSaCS 2011. LNCS, vol. 6604, pp. 365–380. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19805-2_25

14. Gabbay, M.J., Ghica, D.R., Petrisan, D.: Leaving the nest: nominal techniques for variables with interleaving scopes. In: Computer Science Logic, CSL 2015. LIPIcs, vol. 41, pp. 374–389. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2015)
15. Grigore, R., Tzevelekos, N.: History-register automata. Log. Meth. Comput. Sci. **12**(1), 1–32 (2016)
16. Grumberg, O., Kupferman, O., Sheinvald, S.: Variable automata over infinite alphabets. In: Dediu, A.-H., Fernau, H., Martín-Vide, C. (eds.) LATA 2010. LNCS, vol. 6031, pp. 561–572. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13089-2_47
17. Hennessy, M.: A fully abstract denotational semantics for the pi-calculus. Theor. Comput. Sci. **278**, 53–89 (2002)
18. Kaminski, M., Francez, N.: Finite-memory automata. Theor. Comput. Sci. **134**, 329–363 (1994)
19. Kaminski, M., Tan, T.: Regular expressions for languages over infinite alphabets. Fund. Inform. **69**, 301–318 (2006)
20. Kaminski, M., Zeitlin, D.: Finite-memory automata with non-deterministic reassignment. Int. J. Found. Comput. Sci. **21**, 741–760 (2010)
21. Kozen, D., Mamouras, K., Petrişan, D., Silva, A.: Nominal Kleene coalgebra. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) ICALP 2015. LNCS, vol. 9135, pp. 286–298. Springer, Heidelberg (2015). doi:10.1007/978-3-662-47666-6_23
22. Kozen, D., Mamouras, K., Silva, A.: Completeness and incompleteness in nominal Kleene algebra. In: Kahl, W., Winter, M., Oliveira, J.N. (eds.) RAMICS 2015. LNCS, vol. 9348, pp. 51–66. Springer, Cham (2015). doi:10.1007/978-3-319-24704-5_4
23. Kürtz, K., Küsters, R., Wilke, T.: Selecting theories and nonce generation for recursive protocols. In: Formal methods in Security Engineering, FMSE 2007, pp. 61–70. ACM (2007)
24. Kurz, A., Suzuki, T., Tuosto, E.: On nominal regular languages with binders. In: Birkedal, L. (ed.) FoSSaCS 2012. LNCS, vol. 7213, pp. 255–269. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28729-9_17
25. Libkin, L., Tan, T., Vrgoc, D.: Regular expressions for data words. J. Comput. Syst. Sci. **81**, 1278–1297 (2015)
26. Manuel, A., Muscholl, A., Puppis, G.: Walking on data words. Theor. Comput. Sys. **59**, 180–208 (2016)
27. Neven, F., Schwentick, T., Vianu, V.: Finite state machines for strings over infinite alphabets. ACM Trans. Comput. Log. **5**, 403–435 (2004)
28. Pitts, A.: Nominal Sets: Names and Symmetry in Computer Science. Cambridge University Press, Cambridge (2013)
29. Rutten, J.: Universal coalgebra: a theory of systems. Theor. Comput. Sci. **249**(1), 3–80 (2000)
30. Segoufin, L.: Automata and logics for words and trees over an infinite alphabet. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 41–57. Springer, Heidelberg (2006). doi:10.1007/11874683_3
31. Stockhusen, C., Tantau, T.: Completeness results for parameterized space classes. In: Gutin, G., Szeider, S. (eds.) IPEC 2013. LNCS, vol. 8246, pp. 335–347. Springer, Cham (2013). doi:10.1007/978-3-319-03898-8_28

32. Turner, D., Winskel, G.: Nominal domain theory for concurrency. In: Grädel, E., Kahle, R. (eds.) CSL 2009. LNCS, vol. 5771, pp. 546–560. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04027-6_39
33. Tzevelekos, N.: Fresh-register automata. In: Principles of Programming Languages, POPL 2011, pp. 295–306. ACM (2011)
34. Wysocki, T.: Alternating register automata on finite words. Master's thesis, University of Warsaw (2013). (In Polish)