Nominal Automata with Name Binding

Lutz Schröder\textsuperscript{1}, Dexter Kozen\textsuperscript{2}, Stefan Milius\textsuperscript{1}, and Thorsten Wißmann\textsuperscript{1}

\textsuperscript{1} Friedrich-Alexander-Universität Erlangen-Nürnberg
\textsuperscript{2} Cornell University

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 RNNA\textsubscript{s} 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 RNNA\textsubscript{s} under two notions of freshness: global and local. Under global freshness, RNNA\textsubscript{s} turn out to be equivalent to session automata, and as such have a decidable inclusion problem. Under local freshness, RNNA\textsubscript{s} 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 [29] for a summary). There is a plethora of automata models for data languages [3,16,32], 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 [30] 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.

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 [29,5] 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 $\alpha$-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 $\alpha$-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

---

### Fig. 1. Expressivity of selected data language formalisms (restricted to empty initial register assignment). FSUBAs are properly contained in name-dropping RA.
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 Figure 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 $\alpha$-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 [37] and history-register automata [15], often have decidable emptiness 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. [9,8]. RNNAs and unambiguous RAs are incomparable: Closure under complement implies that the language $L = ‘\text{some letter occurs twice}'$ 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 [27] 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; [30] offers a comprehensive introduction.

**Group actions**  Recall that an action of a group $G$ on a set $X$ is a map $G \times X \rightarrow X$, denoted by juxtaposition or infix $\cdot$, such that $\pi(\rho x) = (\pi \rho)x$ and $1x = x$ for $\pi, \rho \in G$, denoting...
$x \in X$. 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 $f(\pi x) = \pi(f(x))$ 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

\[
\text{fix}\, x = \{\pi \in G \mid \pi x = x\} \quad \text{and} \quad \text{Fix}\, A = \bigcap_{x \in A} \text{fix}\, x.
\]

Note that elements of fix $A$ and 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 Fix $A \subseteq \text{fix}\, x$, and $x$ has finite support if some finite $A$ supports $x$. In this case, there is a least set $\text{supp}(x)$ supporting $x$. We say that $a \in \mathbb{A}$ is fresh for $x$, and write $a \# x$, if $a \notin \text{supp}(x)$. A nominal set is a $G$-set all whose elements have finite support. For every equivariant function $f$ between nominal sets, we have $\text{supp}(f(x)) \subseteq \text{supp}(x)$. The function $\text{supp}$ is equivariant, i.e. $\text{supp}(\pi x) = \pi(\text{supp}(x))$ for $\pi \in G$. Hence $\sharp \text{supp}(x_1) = \sharp \text{supp}(x_2)$ whenever $x_1, x_2$ are in the same orbit of a nominal set $G$. 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 $\text{supp}(S) = \emptyset$); and uniformly finitely supported (ufs) if $\bigcup_{x \in S} \text{supp}(x)$ is finite [36]. We denote by $\mathcal{P}_{\text{fs}}(X)$ and $\mathcal{P}_{\text{ufs}}(X)$ 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 $\text{supp}(S) = \bigcup_{x \in S} \text{supp}(x)$.

**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]

\[
[A]X = (A \times X)/\sim
\]

where $\sim$ abstracts the notion of $\alpha$-equivalence as known from calculi with name binding, such as the $\lambda$-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. $\langle a \rangle \{a, d\} = \{b\} \{b, d\}$ 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 Section 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 \( a \in \mathcal{A} \) as \( \lbrack a \rbrack \), and define an extended alphabet \( \mathcal{A}^* \) by

\[
\mathcal{A}^* = \mathcal{A} \cup \{ \lbrack a \rbrack \mid a \in \mathcal{A} \}.
\]

**Definition 3.1.** A **bar string** is a word over \( \mathcal{A}^* \), i.e. an element of \( \mathcal{A}^* \). The set \( \mathcal{A}^* \) 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 \( \lbrack a \rbrack \). A bar string is **clean** if its bound letters \( \lbrack a \rbrack \) are mutually distinct and distinct from all its free names. We write \( \text{FN}(w) \) for the set of free names of \( w \), and say that \( w \) is **closed** if \( \text{FN}(w) = \emptyset \); otherwise, \( w \) is **open**. We define \( \alpha \)-**equivalence** \( \equiv_\alpha \) on bar strings as the equivalence (not: congruence) generated by \( w\lbrack av \rbrack \equiv_\alpha w\lbrack bu \rbrack \) if \( \langle a \rangle \equiv_\alpha \langle b \rangle \) in \( \mathcal{A}^\alpha \) (Section 2). We write \( [w]_\alpha \) for the \( \alpha \)-equivalence class of \( w \). For a bar string \( w \), we denote by \( \text{ub}(w) \in \mathcal{A}^* \) (for **unbind**) the word arising from \( w \) by replacing all bound names \( \lbrack a \rbrack \) with the corresponding free name \( a \).

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

**Example 3.2.** We have \([ab\lbrack cabb \rbrack]_\alpha \neq [ab\lbrack aab \rbrack]_\alpha = [ab\lbrack ccb \rbrack]_\alpha \neq [ap\lbrack ccp \rbrack]_\alpha\) where \( \text{FN}(ab\lbrack cabb \rbrack) = \text{FN}(ab\lbrack aab \rbrack) = \{a, b\} \). The bar string \( ab\lbrack aab \rbrack \) is not clean, and an \( \alpha \)-equivalent clean one is \( ab\lbrack ccb \rbrack \).

**Definition 3.3.** A **literal language** is a set of bar strings, and a **bar language** is an \( \alpha \)-subset of bar strings modulo \( \equiv_\alpha \), i.e. an \( \alpha \)-subset of

\[
\bar{M} := \mathcal{A}^* / \equiv_\alpha.
\]

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

\[
N(L) = \{ \text{ub}(w) \mid w \text{ clean}, [w]_\alpha \in L \} \subseteq \mathcal{A}^*
\]

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

\[
D(L) = \{ \text{ub}(w) \mid [w]_\alpha \in L \} \subseteq \mathcal{A}^*.
\]

That is, \( D(L) \) is obtained by taking all representatives of \( \alpha \)-equivalence classes in \( L \) and then removing bars, while \( N \) takes only clean representatives. Intuitively, \( D \) enforces local freshness by blocking \( \alpha \)-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 \( \alpha \)-equivalence classes) \( D(\{\lbrack a \rbrack b, \lbrack aa \rbrack\}) = \mathcal{A}^2 = D(\{\lbrack a \rbrack b\}) \). This is what we mean by our slogan that **local freshness is a quotient of global freshness**.

**Remark 3.4.** Again omitting \( \alpha \)-equivalence classes, we have \( D(\{\lbrack a \rbrack b\}) = \mathcal{A}^2 \) because \( \lbrack a \rbrack b \equiv_\alpha \lbrack a \rbrack a \). On the other hand, \( D(\{\lbrack a \rbrack ba\}) = \{\lbrack cdc \rbrack \mid c \neq d\} \) because \( \lbrack a \rbrack ba \not\equiv_\alpha \lbrack a \rbrack a \).
\(|a|aa\). We see here that since our local freshness semantics is based on \(\alpha\)-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, Sec. 9.7] for details): Each time the register allocation algorithm needs a register for a variable name \(\langle |a| \rangle\), 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 \(\langle a \text{ and } \alpha \rangle\) for dynamic allocation and deallocation, respectively, of a name \(a\); in this notation, our \(|a|\) corresponds to \(\langle aa \rangle\).

### 4 Regular Bar Expressions

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

**Definition 4.1.** A nondeterministic finite bar automaton, or bar NFA for short, over \(\bar{A}\) is an NFA \(A\) over \(\bar{A}\). We call transitions of type \(q \xrightarrow{a} q\) in \(A\) free transitions and transitions of type \(q \xrightarrow{\alpha} q\) bound transitions. The literal language \(L_0(A)\) of \(A\) is the language accepted by \(A\) as an NFA over \(\bar{A}\). The bar language \(L_\alpha(A) \subseteq \bar{A}\) (see (1)) accepted by \(A\) is defined as

\[
L_\alpha(A) = L_0(A)/\equiv_\alpha.
\]

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

Similarly, a regular bar expression is a regular expression \(r\) over \(\bar{A}\); the literal language \(L_0(r) \subseteq \bar{A}\) 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 \(\text{deg}(r)\) 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 \(a^\prime\), and bound names \(|a|\) as \(a^\circ\)). It defines an \(\bar{A}\)-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 $A$-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 $A^*$ [6].

Example 4.5. The bar language $L = \{ \epsilon, ba, baab, baababba, baababbaab, \ldots \}$ (omitting equivalence classes) is defined by the regular bar expression $(baab)^*(1 + [ba])$ 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 $s \overset{b}{\rightarrow} t \overset{a}{\rightarrow} u \overset{b}{\rightarrow} v \overset{b}{\rightarrow} s$. Under global freshness, the closed bar language $[a]L$ defines the language of odd-length words over $A$ 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, $[a]L$ defines the $A$-language consisting of all odd-length words over $A$ 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][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a][a]
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 \(A\) 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 \(A^*\).

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 \(A = (Q, \rightarrow, s, F)\) consisting of

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

such that the following conditions are satisfied:

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

The degree \(\deg(A) = \max\{ \# \text{supp}(q) | q \in Q \}\) of \(A\) is the maximum size of supports of states in \(A\).

Remark 5.2. For readers familiar with universal coalgebra [31], we note that RNNA have a much more compact definition in coalgebraic terms, and in fact we regard the coalgebraic definition as evidence that RNNA 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 \(\gamma : Q \rightarrow FQ\) for the functor \(F\) on Nom given by

\[
FX = 2 \times P_{\text{ufs}}(A \times X) \times P_{\text{ufs}}([A]X),
\]

together with an initial state \(s \in Q\). The functor \(F\) is a nondeterministic variant of the functor \(KX = 2 \times X^A \times [A]X\) whose coalgebras are deterministic nominal automata [21]. Indeed Kozen et al. [21] show that the \(\nu\)-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 \( w \in \hat{A}^* \) 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 \( \text{supp}(q') \cup \{ a \} \subseteq \text{supp}(q) \).
2. If \( q \xrightarrow{ab} q' \) in \( A \) then \( \text{supp}(q') \subseteq \text{supp}(q) \cup \{ a \} \).

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 \( \text{FN}(w) = \text{supp}(\langle[w]_\alpha \rangle) \subseteq \text{supp}(q) \). 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 \( \text{supp}(w) \subseteq \text{supp}(s) \).

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

\[
s() \xrightarrow{a} t(a) \xrightarrow{ab} u(a, b)
\]

(with \( a, b \) ranging over distinct names in \( \hat{A} \)), where \( s() \) is initial and the states \( u(-, -) \) are final, we have \([u]_\alpha \in L_\alpha(A) \) but the \( \alpha \)-equivalent \([u]\alpha \) is not in \( L_\alpha(A) \). Crucially, closure of \( L_\alpha(A) \) under \( \alpha \)-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 \( N \subseteq \text{supp}(q) \) there exists a state \( q|_N \) in \( A \) that restricts \( q \) to \( N \); that is, \( \text{supp}(q|_N) = N \). \( q|_N \) 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. \( q|_N \xrightarrow{\alpha} q' \) whenever \( q \xrightarrow{\alpha} q' \) and \( \text{supp}(q') \cup \{ a \} \subseteq N \), and \( q|_N \xrightarrow{ab} q' \) whenever \( q \xrightarrow{ab} q' \) and \( \text{supp}(q') \subseteq N \cup \{ a \} \).

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 \( \alpha \)-equivalence is restored under name-dropping:

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

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 $n 2^k$ orbits.

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

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

where $q$ is a state in $A$, $N \subseteq \text{supp}(q)$, and $\text{Fix}(N)q$ denotes the orbit of $q$ under $\text{Fix}(N)$. The final states are the $q|_N$ with $q$ final in $A$, and the initial state is $s|_{\text{supp}(s)}$, 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
- $q|_N \xrightarrow{\pi F} q'|_{N'}$ whenever $q \xrightarrow{\pi F} q''$, $N'' \subseteq \text{supp}(q'') \cap (N \cup \{b\})$, and $(\alpha)(q'|_{N'}) = \langle \beta \rangle(q''|_{N''})$.

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

Equivalence to bar NFAs We proceed to show that RNNA is expressively equivalent to bar NFAs by providing mutual translations. In consequence, we obtain a Kleene theorem connecting RNNA 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 $N_q = \text{supp}(L_\alpha(q))$. The set $\bar{Q}$ of states of $\bar{A}$ consists of pairs

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

where $F_N$ abbreviates $\text{Fix}(N)$ and $\pi F_N$ denotes a left coset. Left cosets for $F_N$ can be identified with injective renamings $N \to \bar{N}$; intuitively, $(q, \pi F_N)$ restricts $q$ to $N$ and renames $N$ according to $\pi$. (That is, we construct a configuration space, as in other translations into NOFAs [7,5]: here, we create virtual registers according to $\text{supp}(L_\alpha(q))$.) 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 $(s, F_N)$, 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

$$(q, \pi F_N) \xrightarrow{\pi F} (q', \pi' F_{N'}) \text{ whenever } q \xrightarrow{\pi F} q', N' \subseteq N \cup \{b\}, (a)\pi' F_{N'} = \langle \pi(b) \rangle \pi F_{N'}.$$
We next present the reverse construction, i.e. given an RNNA \( \alpha \), additional states then arise from name dropping; e.g. for \( \alpha \), for \( L_\alpha(q) \) entries that are either (pairwise distinct) names or \( \bot \), indicating that the corresponding name from \( \text{supp}(L_\alpha(q)) \) 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

\[
\begin{align*}
  s(c) & \xrightarrow{a} t(c, b) \xrightarrow{c} u(b) & \text{for } b \neq c, \text{ with } s(c), u(b) \text{ final for all } b, c \in \mathbb{A}, \\
  & \text{and } s(c) \text{ initial.}
\end{align*}
\]

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

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

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

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

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 (Section 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 \( (\mathbb{B}a) \alpha^*(1 + \mathbb{B}a) \) 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 $\rightarrow \subseteq Q \times A \times Q$, where we write $q \xrightarrow{a} p$ for $(q, a, p) \in \rightarrow$. The $A$-language $L(A) = \{w \mid A \text{ accepts } w\}$ accepted by $A$ is defined in the standard way: extend the transition relation to words $w \in A^*$ 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 $\gamma : Q \rightarrow GQ$ for the functor $GX = 2 \times \mathcal{P}_{eq}(A \times X)$ (2 = $\{\top, \bot\}$) on the category $\text{Nom}$ 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 $A$ (Section 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 $A$-languages. We can convert a closed RNNA $A$ into a NOFA $D(A)$ accepting $D(L_\alpha(A))$ by simply replacing every transition $q \xrightarrow{\alpha} q'$ with a transition $q \xrightarrow{\alpha} q'$. We show that the image of this translation is a natural class of NOFAs:

Definition 6.3. A NOFA $A$ is non-spontaneous if $\text{supp}(s) = \emptyset$ for initial states $s$, and $\text{supp}(q') \subseteq \text{supp}(q) \cup \{a\}$ whenever $q \xrightarrow{\alpha} q'$. (In words, $A$ is non-spontaneous if transitions $q \xrightarrow{\alpha} q'$ in $A$ create no new names other than $a$ in $q'$.) Moreover, $A$ is $\alpha$-invariant if $q \xrightarrow{\alpha} q''$ whenever $q \xrightarrow{b} q'$, $b \neq q$, and $\langle a \rangle q'' = \langle b \rangle q'$ (this condition is automatic if $a \neq b$). Finally, $A$ is name-dropping if for each state $q$ and each set $N \subseteq \text{supp}(q)$ of names, there exists a state $q|_N$ that restricts $q$ to $N$, i.e. $\text{supp}(q|_N) = N$, $q|_N$ is final if $q$ is final, and

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

**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 $\alpha$-invariant.

**Proposition 6.5.** For every non-spontaneous and name-dropping NOFA, there is an equivalent non-spontaneous, name-dropping, and $\alpha$-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. \(\Box\)

**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 [18,29]; 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 [18,38]. Name-dropping restricts expressivity further, as witnessed by the language \{ab | a \neq b\} mentioned above. In return, it buys decidability of inclusion (Section 7), while for non-spontaneous NOFAs even universality is undecidable [5,29]. 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 $(b)^*a(b)^*a$ 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* $\phi$. Register assignments $w \in R := (\mathcal{A} \cup \{\bot\})^n$ determine the, possibly undefined, contents of the $n$ registers, and *configurations* are elements of $C \times R$. Transition constraints are equivariant subsets $\phi \subseteq C \times R \times R$, 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 \( N \subseteq A \) 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:

- \( \text{cmp}_i = \{(w, a, v) \mid w_i = a\} \) (block unless register \( i \) contains the input)
- \( \text{store}_i = \{(w, a, v) \mid v_i \in \{\bot, a\}\} \) (store the input in register \( i \) or forget)
- \( \text{fresh}_i = \{(w, a, v) \mid a \neq w_i\} \) (block if register \( i \) contains the input)
- \( \text{keep}_{ji} = \{(w, a, v) \mid v_i \in \{\bot, w_j\}\} \) (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 \( c_0 \xrightarrow{\text{store}_1} c_1 \xrightarrow{\text{fresh}_1 \land \text{keep}_{11}} c_2 \xrightarrow{\text{cmp}_1} c_3 \), 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 \( \text{cmp}_1 \).

## 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 \( \text{EXPSPACE} \). For global freshness, this essentially just rephrases 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 \( \text{EXPSPACE} \); 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 \( \deg(A_2) \log(\deg(A_1) + \deg(A_2) + 1) \).

The theorem can be rephrased as saying that bar language inclusion of NFA is in parametrized polynomial space (para-\( \text{PSPACE} \)) [34], 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 \( \text{NEXPSPACE} \) 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 \( \Xi \) of states in the name-dropping RNNAs \( A_2 \) generated by \( A_2 \) as described in Construction 5.10, with \( q \) initialized to \( s_1 \) and \( \Xi \) to \( \{ (s_2, \text{id} F_{N_{s_2}}) \} \). 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** \( \Xi' \) of all states of \( A_2 \) reachable from states in \( \Xi \) via \( \alpha \)-transitions (literally, i.e. not up to \( \alpha \)-equivalence) and update \( \Xi \) to \( \Xi' \).

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 \( \Xi \) 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 $\pi F_N$ can be represented as injective renamings $N \to \mathbb{A}$. Note that $\Xi$ will only ever contain states $(q, \pi F N)$ such that the image $\pi N$ 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, $id N_2$ consists only of names literally occurring in $A_2$, and the only names that are added are those occurring in transitions guessed in Step 1, i.e. occurring literally in $A_1$. So states $(q, \pi F N)$ in $\Xi$ can be coded using partial functions $N q \mapsto P$. Since $\# P \leq \deg(A_1) + \deg(A_2)$, there are at most $k \cdot (\deg(A_1) + \deg(A_2) + 1)^{\deg(A_2)} = k \cdot 2^{\log(\deg(A_1) + \deg(A_2) + 1)}$ 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 $\mathbb{A}^*$ generated by $w \subseteq w'$. We denote by $\mathbb{A}^*$ generated by $w \subseteq w'$. We denote by $\mathbb{A}^*$ generated by $w \subseteq w'$.

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 $w' \sqsubseteq w$ 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 $\deg(A_2) \log(\deg(A_1) + \deg(A_2) + 1)$.

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

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, MA, USA (1986)
2. Bielecki, M., Hidders, J., Paredaens, J., Tyszkiewicz, J., den Bussche, J.V.: Navigating with a browser. In: Automata, Languages and Programming, ICALP 2002. LNCS, vol. 2380, pp. 764–775. Springer (2002)
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 (November 2013), draft book; available at 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 (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 (2014)
7. Ciancia, V., Tuosto, E.: A novel class of automata for languages on infinite alphabets. Tech. rep., University of Leicester (2009), cS-09-003
8. Colcombet, T., Puppis, G., Skrypczak, M.: Unambiguous register automata, preprint
9. Colcombet, T.: Unambiguity in automata theory. In: Descriptional Complexity of Formal Systems, DCFS 2015. LNCS, vol. 9118, pp. 3–18. Springer (2015)
10. Demri, S., Lazic, R.: LTL with the freeze quantifier and register automata. ACM Trans. Comput. Log. 10 (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: Foundations of Software Science and Computational Structures, FOSSACS 2011. LNCS, vol. 6604, pp. 365–380. Springer (2011)
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) (2016)
16. Grumberg, O., Kupferman, O., Sheinvald, S.: Variable automata over infinite alphabets. In: Language and Automata Theory and Applications, LATA 2010. LNCS, vol. 6031, pp. 561–572. Springer (2010)
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., Petrisan, D., Silva, A.: Nominal Kleene coalgebra. In: Automata, Languages, and Programming, ICALP 2015. LNCS, vol. 9135, pp. 286–298. Springer (2015)
22. Kozen, D., Mamouras, K., Silva, A.: Completeness and incompleteness in nominal kleene algebra. In: Relational and Algebraic Methods in Computer Science, RAMiCS 2015. LNCS, vol. 9348, pp. 51–66. Springer (2015)
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: Foundations of Software Science and Computational Structures, FOSSACS 2012. LNCS, vol. 7213, pp. 255–269. Springer (2012)
25. Libkin, L., Tan, T., Vrgoc, D.: Regular expressions for data words. J. Comput. Syst. Sci. 81, 1278–1297 (2015)
26. MacLane, S.: Categories for the working mathematician. Springer, 2nd edn. (1998)
27. Manuel, A., Muscholl, A., Puppis, G.: Walking on data words. Theory Comput. Syst. 59, 180–208 (2016)
28. Milius, S., Schröder, L., Wißmann, T.: Regular behaviours with names: On rational fixpoints of endofunctors on nominal sets. Appl. Cat. Struct. (2016), in print
29. Neven, F., Schwentick, T., Vianu, V.: Finite state machines for strings over infinite alphabets. ACM Trans. Comput. Log. 5, 403–435 (2004)
30. Pitts, A.: Nominal Sets: Names and Symmetry in Computer Science. Cambridge University Press (2013)
31. Rutten, J.: Universal coalgebra: a theory of systems. Theoret. Comput. Sci. 249(1), 3–80 (2000)
32. Segoufin, L.: Automata and logics for words and trees over an infinite alphabet. In: Computer Science Logic, CSL 2006. LNCS, vol. 4207, pp. 41–57. Springer (2006)
33. Silva, A., Bonchi, F., Bonsangue, M.M., Rutten, J.J.M.M.: Generalizing determinization from automata to coalgebras. Log. Methods Comput. Sci 9(1:9) (2013)
34. Stockhusen, C., Tantau, T.: Completeness results for parameterized space classes. In: Parameterized and Exact Computation, IPEC 2013. LNCS, vol. 8246, pp. 335–347. Springer (2013)
35. Tal, A.: Decidability of Inclusion for Unification Based Automata. Master’s thesis, Technion (1999)
36. Turner, D., Winskel, G.: Nominal domain theory for concurrency. In: Computer Science Logic, CSL 2009. pp. 546–560 (2009)
37. Tzevelekos, N.: Fresh-register automata. In: Principles of Programming Languages, POPL 2011. pp. 295–306. ACM (2011)
38. Wysocki, T.: Alternating register automata on finite words. Master’s thesis, University of Warsaw (2013). (In Polish)
A Omitted Proofs

In this appendix we assume that readers are familiar with basic notions of category theory (see e.g. [26]), with the theory of algebras and coalgebras for a functor (see e.g. [31]), and with basic properties of nominal sets (see e.g. [30]).

A.1 Abstraction in Nominal Sets

We occasionally use, without express mention, the following alternative description of equality in the abstraction $[A]X$, which formalizes the usual intuitions about $\alpha$-equivalence:

**Lemma A.1.** Let $a, b \in A$ and $x, y \in X$. Then $\langle a \rangle x = \langle b \rangle y$ in $[A]X$ iff either

(i) $(a, x) = (b, y)$, or
(ii) $b \neq a$, $b \not\equiv x$, and $(ab) \cdot x = y$.

**Proof.** 'If': the case where (i) holds is trivial, so assume (ii). Let $c$ be fresh; we have to show $(ca) \cdot x = (cb) \cdot y$. But $(cb) \cdot y = (cb) \cdot (ab) \cdot x = (ca) \cdot x$, where we use in the last step that $b, c$ are both fresh for $x$ so that $(ca)(acb) = (ca)(abc) = (bc)$ fixes $x$.

'Only if': We assume $(a, x) \neq (b, y)$ and prove (ii). We first show $a \neq b$: Assume the contrary. Let $c$ be fresh; by the definition of abstraction, we then have $(ca) \cdot x = (cb) \cdot y$, so $y = (cb)(ca) \cdot x = (ca)(ca) \cdot x = x$, contradiction. We have supp$(x) \subseteq \{a\} \cup$ supp$(\langle a \rangle x) = \{a\} \cup$ supp$(\langle b \rangle y)$, whence $b \not\equiv x$ since $a \neq b$ and $b \not\equiv \langle b \rangle y$. Finally, with $c$ as above $y = (cb)^{-1}(ca) \cdot x = (cb)(ca) \cdot x = (acb) \cdot x = (ab) \cdot x$, again because $(ab)^{-1}(acb) = (ab)(abc) = (bc)$ and $b, c$ are fresh for $x$.

As an easy consequence we obtain:

**Corollary A.2.** Let $X$ be a nominal set, $a \in A$ and $x \in X$. Then supp$(\langle a \rangle x) =$ supp$(x) - \{a\}$.

**Proof of Lemma 2.2** Firstly, any finite set $S \subseteq X$ is ufs, because $\bigcup_{y \in S}$ supp$(y)$ is a finite union of finite sets. Secondly, for any ufs $S \subseteq X$, we have supp$(S) = \bigcup_{y \in S}$ supp$(y)$, which is a finite union (because $X$ is orbit-finite) of again finite sets.

A.2 Proofs and Lemmas for Section 3

**Lemma A.3.** The operator $N$ is injective on closed bar languages.

The proof of Lemma A.3 relies on the following simple fact:

**Lemma A.4.** The operator $ub$ is injective on closed clean bar strings.
Proof. Let $w_1, w_2$ be closed clean bar strings, and let $\text{ub}(w_1) = \text{ub}(w_2)$. Assume for a contradiction that $w_1 \neq w_2$. Picking the leftmost position where $w_1$ and $w_2$ differ, we have w.l.o.g. $u, v_1, v_2 \in \bar{\mathbb{A}}$ and $a \in \mathbb{A}$ such that

$$w_1 = uav_1 \quad \text{and} \quad w_2 = uav_2.$$ 

Since $w_1$ is closed, $u$ must contain $\llbracket a \rrbracket$, in contradiction to $w_2$ being clean. \qed

Proof (Lemma A.3). Let $L_1, L_2$ be closed bar languages such that $N(L_1) = N(L_2)$, and let $[w]_\alpha \in L_1$. We have to show $[w]_\alpha \in L_2$. We have that $w$ is closed, and w.l.o.g. $w$ is clean. Then $\text{ub}(w) \in N(L_1)$, and hence $\text{ub}(w) \in N(L_2)$, so there exists a clean and closed $w'$ such that $\text{ub}(w') = \text{ub}(w)$ and $[w']_\alpha \in L_2$. By Lemma A.4, $w = w'$, so that $[w]_\alpha = [w']_\alpha \in L_2$ as required. \qed

### A.3 Proofs and Lemmas for Section 5

**Definition A.5.** Given a state $q$ in an RNNA $A$ we write $L_0(q)$ and $L_\alpha(q)$ for the literal language and the bar language, respectively, accepted by the automaton obtained by making $q$ the initial state of $A$.

**Lemma A.6.** In an RNNA, the map $q \mapsto L_\alpha(q)$ is equivariant.

Proof. Note first that the set of bar strings $\bar{\mathbb{A}}^*$ is the initial algebra for the functor $S X = 1 + \mathbb{A} \times X + \mathbb{A} \times X$ on Nom. And the set $\bar{M}$ of bar strings modulo $\alpha$-equivalence is the initial algebra for the functor $S_\alpha X = 1 + \mathbb{A} \times X + \llbracket \mathbb{A} \rrbracket X$ on Nom. The functor $S_\alpha$ is a quotient of the functor $S$ via the natural transformation $q : S \to S_\alpha$ given by the canonical quotient maps $\mathbb{A} \times X \to \llbracket \mathbb{A} \rrbracket X$. The canonical quotient map $[-]_\alpha : \bar{\mathbb{A}}^* \to \bar{M}$ that maps every bar string to its $\alpha$-equivalence class is obtained inductively, i.e. $[-]_\alpha$ is the unique equivariant map such that the following square commutes:

$$
\begin{array}{ccc}
S\bar{\mathbb{A}}^* & \xrightarrow{\iota} & \bar{\mathbb{A}}^* \\
S[-]_\alpha \downarrow & & \downarrow [-]_\alpha \\
S\bar{M} & \xrightarrow{\iota_\alpha} & \bar{M}
\end{array}
$$

where $\iota : S\bar{\mathbb{A}}^* \to \bar{\mathbb{A}}^*$ and $\iota_\alpha : S_\alpha\bar{M} \to \bar{M}$ are the structures of the initial algebras, respectively. Since the map $[-]_\alpha$ is equivariant we thus have $\pi[w]_\alpha = [\pi w]_\alpha$ for every $w \in \bar{\mathbb{A}}^*$.

Now we prove the statement of the lemma. Since both free and bound transitions are equivariant, the literal language $L_0(-)$ is equivariant. It follows that the bar language $L_\alpha(-)$ is equivariant: If $m \in L_\alpha(q)$ then there is $w \in L_0(q)$ such that $[w]_\alpha = m$. For $\pi \in G$, it follows that $\pi \cdot w \in L_0(\pi q)$, and hence $[\pi \cdot w]_\alpha \in L_\alpha(\pi q)$. But $[\pi \cdot w]_\alpha = \pi[w]_\alpha$, so $\pi m \in L_\alpha(\pi q)$. \qed

**Proof of Lemma 5.4**

1. Consider the ufs set $Z = \{(a, q') \mid q \xrightarrow{a} q'\}$. Then we have $\text{supp}(q') \cup \{a\} = \text{supp}(a, q') \subseteq \text{supp}(Z) \subseteq \text{supp}(q)$ where the second inclusion holds because $Z$ is ufs, and the third because $Z$ depends equivariantly on $q$. 


2. Consider the ufs set \( Z = \{ [a]q' \mid q \xrightarrow{\text{L}} q' \} \). Then we have \( \text{supp}(q') \subseteq \text{supp}( [a]q' ) \cup \{ a \} \subseteq \text{supp}(Z) \cup \{ a \} \subseteq \text{supp}(q) \cup \{ a \} \) where the second inclusion holds because \( Z \) is ufs, and the third because \( Z \) depends equivariantly on \( q \).

**Remark A.7.** Given an RNNA \( A \) with the state set \( Q \) the paths in \( A \) form the initial algebra for the functor \( Q \times S(\cdot) \), where \( S \) is the functor in the proof of Lemma A.6. Paths in \( A \) modulo \( \alpha \)-equivalence then form the initial algebra for \( Q \times S_{\alpha}(\cdot) \) and the canonical quotient map \([\cdot]_{\alpha} \) mapping a path to its \( \alpha \)-equivalence class is obtained by initiality similarly as the canonical quotient map in Lemma A.6.

**Proof of Lemma 5.7.**

**Lemma A.8.** Let \( A \) be a name-dropping RNNA, and let \( q|_{N} \) restrict a state \( q \) in \( A \) to \( N \subseteq \text{supp}(q) \). Then \( \{ w \in L_{0}(q) \mid \text{FN}(w) \subseteq N \} \subseteq L_{0}(q|_{N}) \).

**Proof.** Induction on the length of \( w \in L_{0}(q) \) with \( \text{FN}(w) \subseteq N \), with the base case immediate from the finality condition in Definition 5.6. So let \( w = \alpha v \) with \( \alpha \in \bar{A} \), accepted via a path \( q \xrightarrow{\alpha} q' \xrightarrow{\epsilon} p \), and let \( q'|_{N_v} \) restrict \( q' \) to \( N_v := \text{FN}(v) \). By the induction hypothesis, \( v \in L_{0}(q|_{N_v}) \). Moreover, \( q \xrightarrow{\alpha} q'|_{N_v} \). We are done once we show that \( q|_{N} \xrightarrow{\alpha} q'|_{N_v} \). If \( \alpha \) is free, then we have to show \( N_v \cup \{ \alpha \} \subseteq N \), and if \( \alpha = \text{fsuc} \) is bound, we have to show \( N_v \subseteq N \cup \{ \alpha \} \). In both cases, the requisite inclusion is immediate from \( \text{FN}(\alpha v) \subseteq N \).

**Proof (Lemma 5.7).** We use induction on the word length. It suffices to show that \( L_{0}(A) \) is closed under single \( \alpha \)-conversion steps. So let \( v|aw \in \bar{A}^{*} \in L_{0}(A) \), via a path \( s = q_{0} \xrightarrow{a} q_{1} \xrightarrow{b} q_{2} \xrightarrow{w} q_{3} \) (with \( q_{3} \) final), let \( b \neq a \) with \( b \notin \text{FN}(w) \), and let \( w' \) be obtained from \( w \) by replacing free occurrences of \( a \) with \( b \). We have to show that \( v|bw' \in L_{0}(A) \); it suffices to show that \( bw' \in L_{0}(q_{1}) \). Put \( N = \text{supp}(q_{2}) - \{ b \} \), and let \( q_{2}|_{N} \) restrict \( q_{2} \) to \( N \). By Lemma A.8, \( w \in L_{0}(q_{2}|_{N}) \), and hence \( [w]_{\alpha} \in L_{0}(q_{2}|_{N}) \). By Lemma A.6, it follows that \( [w']_{\alpha} \in L_{\alpha}((ab)(q_{2}|_{N})) \), so by the induction hypothesis, \( w' \in L_{0}((ab)(q_{2}|_{N})) \). We clearly have \( q_{1} \xrightarrow{b} q_{2}|_{N} \), and by \( \alpha \)-invariance, \( q_{1} \xrightarrow{a} (ab)(q_{2}|_{N}) \) because \( b \neq (q_{2}|_{N}) \). Thus, \( bw' \in L_{0}(q_{1}) \) as required.

**Proof of Lemma 5.8.**

**Definition A.9.** Given the transition data of an RNNA \( A \) (not necessarily assuming any finiteness and invariance conditions) and a state \( q \) in \( A \), we denote by \( \text{fsuc}(q) \) the set

\[
\text{fsuc}(q) = \{ (a, q') \mid q \xrightarrow{\alpha} q' \}
\]

of free transitions of \( q \), and by \( \text{bsuc}(q) \) the set

\[
\text{bsuc}(q) = \{ (a)q' \mid q \xrightarrow{b} q' \}
\]

of bound transitions of \( q \) modulo \( \alpha \)-equivalence.
Note that under $\alpha$-invariance of transitions we have $(a)q' \in \text{bsuc}(q)$ if and only if $q \xrightarrow{\alpha} q'$.

Before we proceed to the proof of the lemma we note the following general fact about nominal sets: for the value of $\pi \cdot x$, it matters only what $\pi$ does on the atoms in $\text{supp}(x)$:

**Lemma A.10.** For $x \in (X, \cdot)$ and any $\pi, \sigma \in G$ with $\pi(v) = \sigma(v)$ for all $v \in \text{supp}(x)$, we have $\pi \cdot x = \sigma \cdot x$.

**Proof.** Under the given assumptions, $\pi^{-1}, \pi \in \text{Fix}(\text{supp}(x)) \subseteq \text{fix}(x)$. \hfill $\square$

**Proof (Lemma 5.8).** Let $A$ be an RNNA with set $Q$ of states.

(1) We construct an equivalent name-dropping RNNA $A'$ as follows. As states, we take pairs

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

where $q \in Q$, $N \subseteq \text{supp}(q)$, and $\text{Fix}(N)q$ denotes the orbit of $q$ under $\text{Fix}(N)$. We define an action of $G$ on states by $\pi \cdot (q|_N) = (\pi q)|_N$. To see well-definedness, let $\pi' \in \text{Fix}(N)$ (i.e. $(\pi'q)|_N = q|_N$); we have to show $(\pi'q)|_N = (\pi q)|_N$. Since $(\pi\pi'\pi^{-1})q = \pi\pi'q$, this follows from $\pi\pi'\pi^{-1} \in \text{Fix}(\pi N)$. The map $(q, N) \mapsto q|_N$ is equivariant, which proves the bound on the number of orbits in $A'$. A state $q|_N$ is final if $q$ is final in $A$; this clearly yields an equivariant subset of states of $A'$. The initial state of $A'$ is $s|_{\text{supp}(s)}$ where $s$ is the initial state of $A$. We have

$$\text{supp}(q|_N) = N;$$

in particular, the states of $A'$ form a nominal set. To see ‘$\subseteq$’ in (5), it suffices to show that $N$ supports $q|_N$. So let $\pi \in \text{Fix}(N)$. Then $\pi \cdot (q|_N) = (\pi q)|_N = q|_N$, as required. For ‘$\supseteq$’, let $a \in N$; we have to show that $N - \{a\}$ does not support $q|_N$. Assume the contrary. Pick $b \neq q$. Then $(ab) \in \text{Fix}(N - \{a\})$, so $(ab) \cdot (q|_N) = (ab) \cdot (\pi q)|_N = \text{Fix}(N)q = q|_N$. In particular, $q \in \text{Fix}(ab) \cdot N \cdot (ab) \cdot q$, i.e. there is $\rho \in \text{Fix}(ab) \cdot N$ such that $\rho(ab) \cdot q = q$. By equivariance of $\text{supp}$, it follows that $\rho(ab) \cdot \text{supp}(q) = \text{supp}(q)$. Now $b \in (ab) \cdot N$, so $\rho(b) = b$. Since $a \in \text{supp}(q)$, it follows that $b \in \rho(ab) \cdot \text{supp}(q)$; but $b \notin \text{supp}(q)$, contradiction.

As transitions of $A'$, we take

- $q|_N \xrightarrow{\alpha} q'|_{N'}$ whenever $q \xrightarrow{\alpha} q'$, $N' \subseteq N$, and $a \in N$, and
- $q|_N \xrightarrow{b} q''|_{N''}$ whenever $q \xrightarrow{b} q''$, $N'' \subseteq \text{supp}(q'') \cap (N \cup \{b\})$, and $(a)(q'|_{N'}) = (b)(q''|_{N''})$.

(We do not require the converse implications. E.g. $q|_N \xrightarrow{\alpha} q'|_{N'}$ need not imply that $q \xrightarrow{\alpha} q'$, only that $\pi q \xrightarrow{\alpha} q'$ for some $\pi \in \text{Fix}(N)$; see also (6) below.) Transitions are clearly equivariant. Moreover, bound transitions are, by construction, $\alpha$-invariant.

**Fact A.11.** By construction, every bound transition in $A'$ is $\alpha$-equivalent to one of the form $q|_N \xrightarrow{b} q'|_{N'}$ where $q \xrightarrow{b} q'$ and $N' \subseteq \text{supp}(q') \cap (N \cup \{a\})$.\footnote{Recall that a transition $q \xrightarrow{b} q'$ is $\alpha$-equivalent to a transition $r \xrightarrow{b} r'$ if $q = r$ and $(a)q' = (b)r'$.}
(2) To see ufs branching, let \( q |_N \) be a state in \( A' \). For free transitions, we have to show that the set

\[
f_{\text{suc}}(q |_N) = \{ (a, q' |_{N'}) \mid N' \subseteq N, a \in N, \pi q \xrightarrow{a} q' \text{ for some } \pi \in \text{Fix}(N) \}
\]

of free successors of \( q |_N \) is ufs. But for \( \pi \in \text{Fix}(N) \), \( N' \subseteq N \), and \( a \in N \), we have \( \pi q \xrightarrow{a} q' \) iff \( q \xrightarrow{a} \pi^{-1} q' \), and then moreover \( \pi^{-1} q' = \text{Fix}(N')q' \), i.e. \( q' |_{N'} = (\pi^{-1} q') |_{N'} \). We thus have

\[
f_{\text{suc}}(q |_N) = \{ (a, (\pi^{-1} q') |_{N'}) \mid N' \subseteq N, a \in N, q \xrightarrow{a} \pi^{-1} q' \} = \{ (a, q' |_{N'}) \mid N' \subseteq N, a \in N, q \xrightarrow{a} q' \},
\]

which is ufs.

We proceed similarly for the bound transitions: We need to show that the set \( b_{\text{suc}}(q |_N) \) of bound successors of \( q |_N \) is ufs. By Fact A.11, a bound transition \( q |_N \xrightarrow{\alpha} q' \) arises from \( \pi \in \text{Fix}(N) \) and \( N' \subseteq \text{supp}(q') \cap (N \cup \{ a \}) \) such that \( \pi q \xrightarrow{\alpha} q' \). Then \( q \xrightarrow{\langle (\pi^{-1} a), \pi^{-1} q' \rangle} \pi^{-1} q' \).

Moreover, we claim that

\[
\langle a \rangle (q' |_{N'}) = \langle \pi^{-1} a \rangle (\pi^{-1} (q' |_{N'})).
\]

To see (7), we distinguish two cases: If \( \pi^{-1} (a) = a \) then the two sides are equal because \( \pi^{-1} \) fixes the support of \( q' |_{N'} \). If \( \pi^{-1} (a) \neq a \) then \( \pi^{-1} a \notin N \) because \( \pi^{-1} \) fixes \( N \), so \( \pi^{-1} a \notin N \cup \{ a \} \) and therefore \( \pi^{-1} a \notin N' = \text{supp}(q' |_{N'}) \). This means that we can \( \alpha \)-equivalently rename \( a \) into \( \pi^{-1} a \) in \( \{ a, q' |_{N'} \} \); since \( \pi^{-1} \) fixes \( N \), the result of this renaming equals \( \{ \pi^{-1} a, \pi^{-1} q' |_{N'} \} \). Since \( \pi^{-1} (q' |_{N'}) = (\pi^{-1} q') |_{\pi^{-1} N'} \) and \( \pi^{-1} N' \subseteq \text{supp}(\pi^{-1} q') \cap (N \cup \{ \pi^{-1} a \}) \) (recall \( \pi \in \text{Fix}(N) \)), (7) proves

\[
b_{\text{suc}}(q |_N) = \{ (\pi^{-1} a) (\langle (\pi^{-1} q') |_{\pi^{-1} N'} \rangle) \mid a \in \text{Fix}(N), \pi^{-1} a \subseteq \text{supp}(\pi^{-1} q') \cap (N \cup \{ \pi^{-1} a \}),
\]

\[
q \xrightarrow{\langle (\pi^{-1} a), \pi^{-1} q' \rangle} \pi^{-1} q'
\]

\[
= \{ \langle a \rangle (q' |_{N'}) \mid N' \subseteq \text{supp}(q') \cap (N \cup \{ a \}), q \xrightarrow{\langle(\pi^{-1} a)\rangle} q' \}.
\]

By (8), \( b_{\text{suc}}(q |_N) \) is ufs; indeed, we have \( \text{supp}(\langle a \rangle (q' |_{N'}) ) = N' - \{ a \} \) so the support of every element of \( b_{\text{suc}}(q |_N) \) is a subset of \( N \). (Note that (8) is not the same as Fact A.11, as in (8) we use a fixed representative \( q \) of \( \text{Fix}(N)q \).)

(3) We show next that \( A' \) is name-dropping. So let \( q |_N \) be a state in \( A' \), and let \( N' \subseteq \text{supp}(q |_N) = N \). We show that \( q |_{N'} \) restricts \( q |_N \) to \( N' \). We first establish that \( q |_{N'} \) has at least the same incoming transitions as \( q |_N \). For the free transitions, let \( \pi \in \text{Fix}(N) \), \( q' \xrightarrow{\alpha} \pi q \) and \( a \in N'' \subseteq N \), so that \( q' |_{N''} \xrightarrow{\alpha} (\pi q) |_{N''} = q |_{N''} \). Then also \( \pi \in \text{Fix}(N') \) and \( N' \subseteq N'' \), so \( q' |_{N''} \xrightarrow{\alpha} \pi q |_{N''} = q |_{N'} \) as required. For the bound transitions, let \( \pi \in \text{Fix}(N) \), let \( \langle a \rangle (q |_N) = \langle b \rangle ((ab) \cdot (q |_N)) \), let \( q' \xrightarrow{\langle b \rangle} \pi (ab) \cdot q \) where \( \pi \in \text{Fix}((ab)N) \), and let \( (ab) \cdot N \subseteq N'' \cup \{ b \} \), so that \( q' |_{N''} \xrightarrow{\langle(\pi b)\rangle} q |_{N''} \). We have to show that \( q' |_{N''} \xrightarrow{\langle(\pi b)\rangle} q |_{N''} \).

From \( q' \xrightarrow{\langle b \rangle} \pi (ab) \cdot q \) we have \( q |_{N''} \xrightarrow{\langle b \rangle} ((\pi (ab) q) |_{(ab)N''}) = (((ab)q) |_{(ab)N''}) \), because
(ab)N' \subseteq (ab)N \subseteq N' \cup \{b\} and \pi \in \text{Fix}((ab)N) \subseteq \text{Fix}((ab)N'). If \(b = a\), we are done. So assume \(b \neq a\). Since \(A'\) is \(\alpha\)-invariant, it remains only to show that

\[
\langle b \rangle ((ab)q|_{(ab)N'}) = \langle a \rangle (q|N'),
\]

i.e. that \(b \notin \text{supp}(q|_{N'})\); but since \(b \neq a\) and \(\langle a \rangle (q|N) = \langle b \rangle ((ab) \cdot (q|N))\), we even have \(b \notin \text{supp}(q|_{N}) \supseteq N' \supseteq \text{supp}(q|_{N'})\).

Next, we show that \(q|_{N'}\) has the requisite outgoing transitions. For the free transitions, let \(q|_{N'} \xrightarrow{a} q'|_{M}\) where \(\text{supp}(q'|_{M}) \cup \{a\} = M \cup \{a\} \subseteq N'\). We have to show \(q|_{N'} \xrightarrow{a} \pi q'\) for some \(\pi \in \text{Fix} M\). By construction of \(A'\), \(q|_{N'} \xrightarrow{a} (\pi q')|_{M} = q'|_{M}\), as required. For the bound transitions, we proceed as follows.

By (8), a given outgoing bound transition of \(q|_{N'} \xrightarrow{\text{In}} q'|_{M}\) yields a state \(r|_{S}\) of \(A'\) and \(b \in A\) such that \(\langle a \rangle q'|_{M} = \langle b \rangle r|_{S}, S \subseteq \text{supp}(r) \cap (N \cup \{a\})\) and \(q \xrightarrow{\text{In}} r\).

Now if \(M \subseteq N' \cup \{a\}\) this yields a transition \(q|_{N'} \xrightarrow{\text{In}} q'|_{M}\) by construction of \(A'\); indeed, we already have \(q \xrightarrow{\text{In}} r, S \subseteq \text{supp}(r)\) and \(\langle a \rangle q'|_{M} = \langle b \rangle r|_{S}\), so it remains to show that \(S \subseteq N' \cup \{b\}\). By Lemma A.1, \(\langle a \rangle q'|_{M} = \langle b \rangle r|_{S}\) iff either \(a = b\) and \(q'|_{M} = r|_{S}\) (and the latter yields \(M = S\), thus we are done), or \(a \neq b, a \neq |_{S}\) and \(\langle a \rangle (r|_{S}) = q'|_{M}\). It follows that \(a \notin S\) and \(M = (ba)S\). Since \((ba)S \subseteq N' \cup \{a\}\) we have equivalently \(S \subseteq (ba)N' \cup \{b\}\). This implies \(S \subseteq N' \cup \{b\}\) using that \(a \notin S\).

(4) It remains to show that \(L_\alpha(A') = L_\alpha(A)\). To show \(\subseteq\), we show that \([w]_\alpha \in L_\alpha(q)\) for every state \(q|_{N}\) in \(A'\) and every \(w \in L_\alpha(q|_{N})\), by induction on \(w\): for the empty word, the claim follows from the definition of final states in \(A'\). For \(w = \alpha v\), let \(q|_{N} \xrightarrow{\alpha} q'|_{N'} \xrightarrow{t} t\) be an accepting path in \(A'\). Then we have \([v]_\alpha v \in L_\alpha(q')\) by induction hypothesis and \(\text{FN}(v) \subseteq N'\) by Corollary 5.5. By (6) and (8), we have \(q \xrightarrow{\alpha} \pi q'\) for some \(\pi \in \text{Fix} N'\). It follows that \(\pi v = v\) and therefore \([v]_\alpha = \pi[v]_\alpha \in L_\alpha(q')\) by the equivariance of \(L_\alpha\) (see Lemma A.6). Hence \([\alpha v]_\alpha \in L_\alpha(q)\).

To see \(L_\alpha(A') \supseteq L_\alpha(A)\), it suffices to note that \(A\) is included as a subautomaton in \(A'\) via the map that takes \(q\) to \(q|\text{supp}(q)\), i.e. \(q \xrightarrow{\alpha} q'\) in \(A\) implies \(q|\text{supp}(q) \xrightarrow{\alpha} q'|\text{supp}(q')\) in \(A'\).

**Lemma A.12.** Let \(q\) be a state in a bar NFA; then \(L_\alpha(q)\) is ufs.

**Proof.** The finitely many transitions of \(A\) only mention letters from a finite subset of \(\bar{A}\), and \(\bigcup_{w \in L_\alpha(q)} \text{supp}(w)\) is contained in that finite subset.

**Details for Theorem 5.11**

As indicated in the text, we split the construction into two parts, and first construct a plain RNNA \(A\). The states of \(A\) are pairs

\[
(q, \pi H_q) \quad \text{where } H_q = \text{Fix}(\text{supp}(L_\alpha(q)))
\]

consisting of a state \(q\) in \(A\) and a left coset \(\pi H_q\), where the action of \(G\) is as on \(\bar{A}\):

\[
\pi_1 \cdot (q, \pi_2 H_q) = (q, \pi_1 \pi_2 H_q).
\]
We continue to write $N_q = \text{supp}(L_\alpha(q))$ (note $H_q = F_{N_q}$ in the notation used in the construction of $\tilde{A}$). The initial state of $\tilde{A}$ is $(s, H_s)$ where $s$ is the initial state of $A$; a state $(q, \pi H_q)$ is final in $\tilde{A}$ iff $q$ is final in $A$. Free transitions in $\tilde{A}$ are of the form

$$(q, \pi H_q) \xrightarrow{\pi(a)} (q', \pi H_{q'}) \quad \text{where } q \xrightarrow{a} q' \text{ and } a \in N_q,$$

(where the condition $a \in N_q$ is automatic unless $L_\alpha(q') = \emptyset$) and bound transitions are of the form

$$(q, \pi H_q) \xrightarrow{\text{b}} (q', \pi H_{q'}) \quad \text{where } q \xrightarrow{\text{b}} q' \text{ and } \langle a \rangle \pi H_q = \langle \pi(b) \rangle \pi H_{q'}.$$

**Remark A.13.** 1. Note that by Lemma 2.1, $N_q = \text{supp}(L_\alpha(q)) = \bigcup_{w \in L_\alpha(q)} \text{supp}(w)$, i.e. $N_q$ is the set of names that appear free in some word $w \in L_\alpha(q)$.

2. Observe that $\pi H_q = \pi' H_q$ iff $\pi'(v) = \pi(v)$ for all $v \in N_q$; $\pi H_q = \pi' H_q$ iff $\pi^{-1} \pi' \in H_q$ iff $\pi^{-1} \pi'(v) = v$ for all $v \in N_q$ iff $\pi'(v) = \pi(v)$ for all $v \in N_q$.

3. For a coset $\pi H_q$, we have

$$\text{supp}(q, \pi H_q) = \text{supp}(\pi H_q) = \pi N_q$$

so the set $\tilde{Q}$ of states of $\tilde{A}$ is a nominal set. This is by Item (2): for $\pi' \in G$, we have $\pi' \pi H_q = \pi H_q$ iff $\pi' \pi(a) = \pi(a)$ for all $a \in N_q$ iff $\pi' \in \text{Fix}(\pi N_q)$.

4. Note that $\langle a \rangle \langle \pi H_q \rangle = \langle b \rangle \langle \pi' H_q \rangle$ implies $\langle a \rangle(q, \pi H_q) = \langle b \rangle(q, \pi' H_q)$ since the action of $G$ on states of $\tilde{A}$ is trivial in the first component.

**Remark A.14.** Left cosets for $H_q$ are in one-to-one correspondence with injections $N_q \to \mathbb{A}$. Indeed, in the light of Remark A.13(2) it suffices to prove that every injection $i : N_q \to \mathbb{A}$ can be extended to a finite permutation. Define $\pi$ by

$$\pi(a) = \begin{cases} i(a) & a \in N_q \\ i^{-n}(a) & \text{else, for } n \geq 0 \text{ minimal s.t. } i^{-n}(a) \notin \{N_q\} \end{cases}$$

For the proof that $\pi$ is indeed a finite permutation see [28, Corollary 2.4].

Transitions from a given state $(q, \pi H_q)$ can be characterized as follows.

**Lemma A.15.** Let $(q, \pi H_q)$ be a state in $\tilde{A}$. Then

$$\text{fsuc}(q, \pi H_q) = \{(\pi(a), (q', \pi H_{q'})) \mid q \xrightarrow{a} q', a \in N_q\} \quad (10)$$

and

$$\text{bsuc}(q, \pi H_q) = \{(\langle a \rangle(q', \pi H_{q'})) \mid q \xrightarrow{\text{b}} q'\}. \quad (11)$$

**Proof.** For the free transitions, we have by definition

$$\text{fsuc}(q, \pi H_q) = \{(\pi'(a), (q', \pi' H_{q'})) \mid \pi' H_q = \pi H_q, q \xrightarrow{a} q', a \in N_q\}.$$
The key ingredient in the proof that \( \tilde{\pi} \) which we now prove. By Remark A.13(2) we know that \( N \) (as \( q \) if \( \pi \) normalization result on paths that uses an obvious notion of \( \alpha \) \( \supp \)). Otherwise, \( \alpha \) Lemma A.17. \[ \langle \pi'(a) \rangle(q', \pi'H_q) = \langle \pi(a) \rangle(q', \pi'H_q), \]

which we now prove. By Remark A.13(2) we know that \( \pi \) and \( \pi' \) agree on \( N_q \). In order to prove (12), we distinguish two cases: if \( \pi(a) = \pi'(a) \) then \( \pi \) and \( \pi' \) agree on \( N_q \subseteq N_q \cup \{a\} \), i.e. \( \pi'H_q = \pi'H_q \), so the two sides of (12) are literally equal. Otherwise, \( a \notin N_q \), and \( \pi', \pi \) differ on \( N_q \) only w.r.t. their value on \( a \). It follows that \( (\pi(a) \pi'(a)) \pi \) and \( \pi' \) agree on \( N_q \subseteq \pi N_q \cup \{a\} = \pi' N_q \cup \pi(a), \) and \( \pi'(a) \notin \pi' N_q \cup \pi(a) \) because \( a \notin N_q \) and \( \pi'(a) \neq \pi(a) \).

The key ingredient in the proof that \( \tilde{A} \) accepts the same bar language as \( A \) will be a normalization result on paths that uses an obvious notion of \( \alpha \)-equivalence on paths in an RNNA (see Remark A.7); explicitly:

**Definition A.16.** \( \alpha \)-equivalence of paths in an RNNA is defined inductively by

\[ q_0 \rightarrow a_1 \rightarrow q_1 \rightarrow a_2 \rightarrow q_2 \rightarrow \cdots \rightarrow a_n \rightarrow q_n \]

is \( \alpha \)-equivalent to

\[ q_0 \rightarrow a'_1 \rightarrow q'_1 \rightarrow a'_2 \rightarrow q'_2 \rightarrow \cdots \rightarrow a'_n \rightarrow q'_n \]

if \( q_1 \rightarrow a_2 \rightarrow q_2 \rightarrow \cdots \rightarrow a_n \rightarrow q_n \) is \( \alpha \)-equivalent to \( q'_1 \rightarrow a'_2 \rightarrow q'_2 \rightarrow \cdots \rightarrow a'_n \rightarrow q'_n \).

\[ q_0 \rightarrow a_1 \rightarrow q_1 \rightarrow a_2 \rightarrow q_2 \rightarrow \cdots \rightarrow a_n \rightarrow q_n \]

is \( \alpha \)-equivalent to

\[ q_0 \rightarrow a'_1 \rightarrow q'_1 \rightarrow a'_2 \rightarrow q'_2 \rightarrow \cdots \rightarrow a'_n \rightarrow q'_n \]

if \( \alpha \{a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow \cdots \rightarrow a_n \rightarrow q_n \} = (\alpha \{b_1 \rightarrow b_2 \rightarrow b_3 \rightarrow \cdots \rightarrow b_n \rightarrow q_n \})_\alpha \), where we use \([\cdot]_\alpha \) to denote \( \alpha \)-equivalence classes of paths.

**Lemma A.17.** The set of paths of an RNNA is closed under \( \alpha \)-equivalence.

**Proof.** Observe that by equivariance, \( G \) acts pointwise on paths. It suffices to show closure under single \( \alpha \)-conversion steps. So let \( q_0 \rightarrow a_1 \rightarrow q_1 \rightarrow a_2 \rightarrow q_2 \rightarrow \cdots \rightarrow a_n \rightarrow q_n \) be path in an RNNA \( A \), denote the path from \( q_1 \) onwards by \( P \), and let \( \langle a \rangle P = \langle b \rangle P' \), so \( P' = (ab) \cdot P \). Then by \( \alpha \)-invariance of \( \rightarrow \), we have \( q_0 \rightarrow (ab)q_1 \), and by equivariance, \((ab) \cdot P\) is a path from \((ab)q_1\).

**Lemma A.18.** Let \( P = q_0 \rightarrow a_1 \rightarrow q_1 \rightarrow a_2 \rightarrow q_2 \rightarrow \cdots \rightarrow a_n \rightarrow q_n \) be a path in an RNNA \( A \), and let \( \langle a \rangle q_1 = \langle b \rangle q'_1 \). Then there exists a path in \( A \) of the form \( q_0 \rightarrow a'_1 \rightarrow q'_1 \rightarrow a'_2 \rightarrow q'_2 \rightarrow \cdots \rightarrow a'_n \rightarrow q'_n \) that is \( \alpha \)-equivalent to \( P \).
Proof. Since $A$ is an RNNA, the support of the $\alpha$-equivalence class of $q_1 \xrightarrow{\alpha_n} \ldots \xrightarrow{\alpha_2} q_n$ is $\text{supp}(q_1)$ (Remark A.19), so we obtain an $\alpha$-equivalent path $q_0 \xrightarrow{\psi} q'_1 \xrightarrow{\alpha'_2} \ldots \xrightarrow{\alpha'_n} q_n$ by renaming $a$ into $b$ in $q_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} q_n$.

Remark A.19. Note that the support of the $\alpha$-equivalence class of a path in an RNNA is the support of its starting state. Indeed, let $[P]_\alpha$ be such an equivalence class and let $q$ be the starting state of $P$. The inclusion $\text{supp}(q) \subseteq \text{supp}([P]_\alpha)$ holds because we have a well-defined equivariant projection from paths to their initial states. The converse inclusion is shown by induction, using Lemma 5.4.

In the proof that $\hat{A}$ accepts $L_\alpha(A)$, the following normalization result for paths is crucial.

Definition A.20. A path in $\hat{A}$ is $\pi$-literal for $\pi \in G$ if all transitions in it are of the form $(q, \pi H_q) \xrightarrow{\pi_1} (q', \pi H_{q'})$ where $\alpha \in \hat{a}$ and $q \xrightarrow{\alpha} q'$.

Intuitively, a $\pi$-literal path is one that uses the same pattern of name reusage for free and bound names as the underlying path in $A$, up to a joint renaming $\pi$ of the free and bound names.

Lemma A.21. Let $P$ be a path in $\hat{A}$ beginning at $(q_0, \pi_0 H_{q_0})$. Then $P$ is $\alpha$-equivalent to a $\pi_0$-literal path.

Proof. We prove the statement by induction over the path length. The base case is trivial. For the inductive step, let $P = (q_0, \pi_0 H_{q_0}) \xrightarrow{\alpha_1} (q_1, \pi_1 H_{q_1}) \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} (q_n, \pi_n H_{q_n})$ be a path of length $n > 0$. If $\alpha_1$ is a free name then $\pi_1 H_{q_1} = \pi_0 H_{q_1}$ by (10); by induction, we can assume that the length-$(n - 1)$ path from $(q_1, \pi_0 H_{q_1})$ onward is $\pi_0$-literal, and hence the whole path is $\pi_0$-literal. If $\alpha_1 = \iota a$ then by (11) we have $(a)(q_1, \pi_1 H_{q_1}) = (\pi_0(b))(q_1, \pi_0 H_{q_1})$ for some transition $q_0 \xrightarrow{\psi} q_1$ in $A$. By Lemma A.18, this induces an $\alpha$-equivalence of $P$ with a path $(q_0, \pi_0 H_{q_0}) \xrightarrow{\iota_0(b)} (q_1, \pi_0 H_{q_1}) \rightarrow \ldots$; by the induction hypothesis, we can transform the length-$(n - 1)$ path from $(q_1, \pi_0 H_{q_1})$ onward into a $\pi_0$-literal one, so that the whole path becomes $\pi_0$-literal as desired.

Lemma A.22. $\hat{A}$ is an RNNA, with as many orbits as $A$ has states, and accepts the bar language $L_\alpha(A)$.

Proof. The free and bound transitions of $\hat{A}$ are equivariant, and the bound transitions are $\alpha$-invariant by construction of the transition relation on $\hat{A}$ (note that all states in the orbit of $(q, \pi H_q)$ have the form $(q, \pi' H_q)$). Every orbit of $\hat{A}$ contains a state of the form $(q, \text{id} H_q)$. This proves the claim on the number of orbits, which implies that $\hat{A}$ is orbit-finite. Finite branching is immediate from Lemma A.15. Thus, $\hat{A}$ is an RNNA.

It remains to show that $L_\alpha(\hat{A}) = L_\alpha(A)$. The inclusion ‘⊆’ is clear because $A$ is a subautomaton in $\hat{A}$ via the inclusion map $f$ taking a state $q$ to $(q, \text{id} H_q)$; i.e. $q \xrightarrow{\alpha} q'$ in $A$ implies $(q, \text{id} H_q) \xrightarrow{\alpha} (q', \text{id} H_{q'})$ in $\hat{A}$. For the reverse inclusion, note that by Lemma A.21, every accepting path of $\hat{A}$ is $\alpha$-equivalent to an $\text{id}$-literal accepting path starting at the initial state $(s, \text{id} H_s)$ of $\hat{A}$; such a path comes from an accepting path in $A$ for the same bar string via the map $f$. 

Nominal Automata with Name Binding

Proof of Theorem 5.11. By combining the above construction of $\tilde{A}$ with that of Lemma 5.8; i.e., we show that $(\tilde{A})'$ is isomorphic, and in fact equal, to $A$: A state in $(\tilde{A})'$ has the form

$$(q, \pi H_q)|_N = (\text{Fix } N) \cdot (q, \pi H_q) = (q, (\text{Fix } N)\pi H_q)$$

for $N \subseteq \text{supp}(q, \pi H_q) = \pi N_q$ (hence $\pi^{-1}N \subseteq N_q$). We claim that

$$(\text{Fix } N)\pi H_q = \pi \text{Fix}(\pi^{-1}N) = \pi F_{\pi^{-1}N}.$$  

(13)

To see ‘$\subseteq$’, let $\rho \in \text{Fix } N$ and $\sigma \in H_q$. Then $\pi^{-1}\rho\pi \in \text{Fix}(\pi^{-1}N)$ and, since $\pi^{-1}N \subseteq N_q$, $\sigma \in \text{Fix}(\pi^{-1}N)$, and $\pi^{-1}\rho\pi\sigma = \pi^{-1}\rho\pi\sigma \in \pi \text{Fix}(\pi^{-1}N)$.

For ‘$\supseteq$’, let $\rho \in \text{Fix}(\pi^{-1}N)$. Then $\pi\rho\pi^{-1} \in \text{Fix}(N)$, so to show $\pi\rho \in \text{Fix}(N)\pi H_q$, it suffices to show $(\pi\rho\pi^{-1})^{-1}\pi\rho \in H_q$. But $(\pi\rho\pi^{-1})^{-1}\pi\rho = \pi \in H_q$.

This proves equality of the state sets. It remains to show that the transitions in $(\tilde{A})'$ and $\tilde{A}$ are the same. The free transitions in $(\tilde{A})'$ are of the form $(q, \pi H_q)|_N \xrightarrow{(\pi(a)} (q', \pi H_{q'})$ where $q \xrightarrow{a} q'$, $N' \subseteq \pi N'_{q'}$, and $N' \cup \{a\} \subseteq N \subseteq \pi N_q$; by (13), they thus have, up to $\alpha$-equivalence, the form $(q, \pi F_{\pi^{-1}N}) \xrightarrow{(\pi(a)} (q', \pi F_{\pi^{-1}N'})$ where $\pi^{-1}N' \subseteq N' \pi^{-1}N' \cup \{a\} \subseteq \pi^{-1}N \subseteq N_q$, and hence are the same as in $\tilde{A}$.

The bound transitions in $(\tilde{A})'$ are, up to $\alpha$-equivalence, those of the form $(q, \pi H_q)|_N \xrightarrow{\text{ Init}(a)} (q', \pi H_{q'})|_{N'}$, where $q \xrightarrow{a} q'$, $N' \subseteq \{a\} \cup N$, $N' \subseteq \pi N_{q'}$; and $N \subseteq \pi N_q$; by (13), they thus have the form $(q, \pi F_{\pi^{-1}N}) \xrightarrow{\text{ Init}(a)} (q', \pi F_{\pi^{-1}N'})$ where $\pi^{-1}N' \subseteq \{a\} \cup \pi^{-1}N$, $\pi^{-1}N' \subseteq N'_{q'}$, and $\pi^{-1}N \subseteq N_q$, and hence again are the same as in $\tilde{A}$.

Proof of Theorem 5.13

We have to show that every accepting path in $A$ is $\alpha$-equivalent to an accepting path in $A_0$. Note that $Q_0$ is closed under free transitions in $A$, so by Lemma A.18, it suffices to show that for every bound transition $q \xrightarrow{a} q'$ in $A$ with $q \in Q_0$ we find an $\alpha$-equivalent transition $q \xrightarrow{a} q''$ in $A_0$. We distinguish the following cases.

- If already $b \in \mathcal{A}_0$ then $\text{supp}(q') \subseteq \text{supp}(q) \cup \{b\} \subseteq \mathcal{A}_0$, so $q' \in Q_0$ and we are done.
- If $b \notin \mathcal{A}_0$ and $b \notin \text{supp}(q')$ then $\text{supp}(q') \subseteq \text{supp}(q) \subseteq \mathcal{A}_0$. In particular, $q'$ is already in $Q_0$ and $*$ is fresh for $q'$, so we can rename $b$ into $*$ and obtain an $\alpha$-equivalent transition $q \xrightarrow{a} q'$ in $A_0$.
- If $b \notin \mathcal{A}_0$ and $b \in \text{supp}(q')$ then $|\text{supp}(q') \cap \mathcal{A}_0| < k$, so that we can pick a name $a \in \mathcal{A}_0$ that is fresh for $q'$. We put $q'' = (ab)q''$; then $b|q'' = \langle a|q''$, and $q'' \in Q_0$ because $\text{supp}(q'') = \{a\} \cup (\text{supp}(q') - \{b\}) \subseteq \mathcal{A}

A.4 Proofs and Lemmas for Section 6

NOFAs as coalgebras We show that the standard description of NOFAs as repeated at the beginning of Section 6 is equivalent to the one as $F$-coalgebras for $FX = 2 \times \ldots \times$. 


For the direction from the standard description to $F$-coalgebras, recall that the transition relation is assumed to be equivariant; therefore, the map taking a state $q$ to $\{(a, q') \mid q \overset{a}{\rightarrow} q'\}$ is equivariant, hence preserves supports and therefore ends up in $FQ$ where $Q$ is the set of states. Conversely, let $\xi : Q \rightarrow FQ$ be an $F$-coalgebra with components $f : Q \rightarrow 2$, $g : Q \rightarrow \mathcal{P}_b(\mathbb{N} \times Q)$. Define the transition relation on $Q$ by $q \overset{a}{\rightarrow} q'$ iff $(a, q') \in g(\xi(q))$, and make $q$ final iff $f(\xi(q)) = \top$. Then finality is equivariant by equivariance of $f$. To see that the transition relation is equivariant let $q \overset{a}{\rightarrow} q'$ and $\pi \in G$. Then $(\pi a, \pi q') \in \pi(g(q)) = g(\pi(q))$ by equivariance of $g$, i.e. $\pi q \overset{\pi a}{\rightarrow} \pi q'$. \hfill \Box

Lemma A.23. If $\pi \in G$ and $q|_N$ restricts a state $q$ in a NOFA to $N \subseteq \text{supp}(q)$, then $\pi(q|_N)$ restricts $\pi q$ to $\pi N$.

Proof. We have $\pi N \subseteq \pi \text{supp}(q) = \text{supp}(\pi q)$, so the claim is well-formed.

For the support of $\pi(q|_N)$, we have $\text{supp}(\pi(q|_N)) = \pi \text{supp}(q|_N) = \pi N$ as required.

By the equivariance of final states we have that $\pi(q|_N)$ is final if $q$ is final.

For incoming transitions, let $p \overset{a}{\rightarrow} \pi q$. Then $\pi^{-1} \cdot \pi^{-1} a \cdot q|_N$ so that $p \overset{a}{\rightarrow} \pi(q|_N)$.

For outgoing transitions, let $\pi q \overset{a}{\rightarrow} q'$ where $\text{supp}(q') \subseteq \pi N \cup \{a\}$. Then $q \overset{\pi^{-1} a}{\rightarrow} \pi^{-1} q'$ by equivariance, and $\text{supp}(\pi^{-1} q') \subseteq \pi N \cup \pi^{-1} a$, so $q|_N \overset{\pi^{-1} a}{\rightarrow} \pi^{-1} q'$ and hence, by equivariance, $\pi(q|_N) \overset{\pi a}{\rightarrow} q'$. \hfill \Box

Proof (Proposition 6.4). In the first claim, ‘only if’ is immediate by Lemma 5.4. To see ‘if’, let $A$ be a non-spontaneous and $\alpha$-invariant NOFA. We construct an RNNA $B$ with the same states as $A$, as follows.

- $q \overset{a}{\rightarrow} q'$ in $B$ iff $q \overset{a}{\rightarrow} q'$ in $A$ and $a \in \text{supp}(q)$.
- $q \overset{1a}{\rightarrow} q'$ in $B$ iff $q \overset{b}{\rightarrow} q''$ in $A$ for some $b, q''$ such that $b \neq q$ and $\langle b \rangle q'' = \langle a \rangle q'$.

The transition relation thus defined is clearly equivariant and $\alpha$-invariant. That for every $q$ the sets $\{(a, q') \mid q \overset{a}{\rightarrow} q'\}$ and $\{\langle a \rangle q' \mid q \overset{1a}{\rightarrow} q'\}$ are ufs (whence finite) easily follows from non-spontaneity.

It remains to verify that $D(B) = A$, i.e. that

$$q \overset{a}{\rightarrow} q' \text{ in } A \iff (q \overset{a}{\rightarrow} q' \text{ or } q \overset{1a}{\rightarrow} q' \text{ in } B).$$

To see the ‘only if’ direction, let $q \overset{a}{\rightarrow} q'$ in $A$. If $a \in \text{supp}(q)$ then $q \overset{a}{\rightarrow} q'$ in $B$. Otherwise, $a \neq q$ and hence $q \overset{1a}{\rightarrow} q'$. For the ‘if’ direction, we have two cases; the case where $q \overset{a}{\rightarrow} q'$ in $B$ is immediate by construction of $B$. So let $q \overset{1a}{\rightarrow} q'$ in $B$, that is, we have $q \overset{b}{\rightarrow} q''$ in $A$ for some $b, q''$ such that $\langle b \rangle q'' = \langle a \rangle q'$ and $b \neq q$. Then by $\alpha$-invariance of $A$, $q \overset{a}{\rightarrow} q'$.

We proceed to prove the second claim, beginning with ‘only if’. So let $C$ be a name-dropping RNNA, let $q$ be a state, let $N \subseteq \text{supp}(q)$, and let $q|_N$ restrict $q$ to $N$ in $C$. We show that $q|_N$ restricts $q$ to $N$ in $D(C)$. The condition $\text{supp}(q|_N) \subseteq N$ is clear, as the
nominal set of states is not changed by $D$. Since $q\mid_N$ has at least the same incoming transitions as $q$ in $C$, the same holds in $D(C)$. For the outgoing transitions, first let $q \xrightarrow{a} q'$ in $D(C)$ where $a \in \text{supp}(q)$ and $\text{supp}(q') \cup \{a\} \subseteq N$. Then either $q \xrightarrow{a} q'$ or $q \xrightarrow{b} q'$ in $C$. In the first case, $q\mid_N \xrightarrow{a} q'$ in $C$ and hence also in $D(C)$. In the second case, we have $\text{supp}(q') \subseteq N \subseteq \{a\} \cup N$ and therefore $q\mid_N \xrightarrow{b} q'$ in $C$, so $q\mid_N \xrightarrow{b} q'$ in $D(C)$. Second, let $q \xrightarrow{a} q'$ in $D(C)$ where $a \neq q$ and $\text{supp}(q') \subseteq N \cup \{a\}$. By Lemma 5.4.1 we know that $q \xrightarrow{a} q'$ in $C$, so $q\mid_N \xrightarrow{a} q'$ in $C$ and hence $q\mid_N \xrightarrow{a} q'$ in $D(C)$.

For the ‘if’ direction of the second claim, let $A$ be a non-spontaneous, name-dropping, and $\alpha$-invariant NOFA. We construct $B$ such that $D(B) = A$ as for the first claim, and show additionally that $B$ is name-dropping. Let $q$ be a state, let $N \subseteq \text{supp}(q)$, and let $q\mid_N$ restrict $q$ to $N$ in $A$. We claim that $q\mid_N$ also restricts $q$ to $N$ in $B$. We first show that $q\mid_N$ has at least the same incoming transitions as $q$ in $B$. For the free transitions, let $p \xrightarrow{a} q$ in $B$. Then by construction of $B$, $p \xrightarrow{a} q$ in $A$ and $a \in \text{supp}(p)$, so since $A$ is name-dropping, $p \xrightarrow{a} q\mid_N$ in $A$ and hence $p \xrightarrow{a} q\mid_N$ in $B$. For the bound transitions, let $p \xrightarrow{b} q$ in $B$, i.e. we have $p \xrightarrow{b} q'$ in $A$ with $b \neq p$ and $(a)q = (b)q'$, in particular $q' = (ab)q$. If $a = b$ then $p \xrightarrow{a} q$ in $A$, and since $A$ is name-dropping $p \xrightarrow{a} q\mid_N$ in $A$ whence in $B$. Otherwise, $b \neq q$. By Lemma A.23, $(ab)(q\mid_N)$ restricts $q'$ to $(ab)N$, so $p \xrightarrow{b} (ab)(q\mid_N)$ in $A$. Since $\text{supp}(q\mid_N) \subseteq \text{supp}(q)$, we have $b \neq q\mid_N$, so $(b)((ab)(q\mid_N)) = (a)(q\mid_N)$. By construction of $B$ we have $q \xrightarrow{a} q\mid_N$ in $B$, as required.

For the outgoing transitions, first let $q \xrightarrow{a} q'$ in $B$ where $\text{supp}(q') \cup \{a\} \subseteq N$. Then $q \xrightarrow{a} q'$ in $A = D(B)$, so $q\mid_N \xrightarrow{a} q'$ in $A$; since $a \in N = \text{supp}(q\mid_N)$, it follows by construction of $B$ that $q\mid_N \xrightarrow{a} q'$ in $B$. Second, let $q \xrightarrow{b} q'$ in $B$ where $\text{supp}(q') \subseteq N \cup \{a\}$. Pick $b \neq (a)q$ (so $b \notin N$); then $(b)((ab)q') = (a)q'$ and therefore $q \xrightarrow{b} (ab)q'$ in $B$ by $\alpha$-invariance. Thus, $q \xrightarrow{b} (ab)q'$ in $A = D(B)$, and $\text{supp}((ab)q') = (ab)\text{supp}(q') \subseteq (ab)(N \cup \{a\}) = N - \{a\} \cup \{b\} \subseteq N \cup \{b\}$, where the last but one equation holds since $b \notin N$. Therefore, $q\mid_N \xrightarrow{b} (ab)q'$ in $A$ since $A$ is name-dropping. By construction of $B$, it follows that $q\mid_N \xrightarrow{b} q'$ in $B$. \[\square\]

Proof (Proposition 6.5). Let $A$ be a non-spontaneous and name-dropping NOFA. We construct a NOFA $\tilde{A}$ by closing $A$ under $\alpha$-equivalence of transitions; that is, $\tilde{A}$ has the same states as $A$ (in particular is orbit-finite), and its transitions are given by

$$q \xrightarrow{a} q' \text{ in } \tilde{A} \text{ iff } q \xrightarrow{a} q' \text{ in } A \text{ or there exist } b, q'' \text{ such that } q \xrightarrow{b} q'' \text{ in } A, b \neq q, \text{ and } (\langle a \rangle q') = (\langle b \rangle q'').$$

We say that a transition $q \xrightarrow{a} q'$ in $\tilde{A}$ is new if it is not in $A$.

**Fact A.24.** If $q \xrightarrow{a} q'$ is new then $a \in \text{supp}(q)$ and there exist $b, q''$ such that $q \xrightarrow{b} q''$ in $A, b \neq q$ (so $a \neq b$), and $(\langle a \rangle q') = (\langle b \rangle q'')$.

We check that $\tilde{A}$ has the requisite properties. First, the transition relation is clearly equivariant. Moreover, $\tilde{A}$ is $\alpha$-invariant by construction.
is non-spontaneous: It suffices to check new transitions $q \xrightarrow{a} q'$. By Fact A.24, we have $a \in \text{supp}(q)$ and $b, q''$ such that $q \xrightarrow{b} q''$ in $A$, $b \not= q$, and $\langle a \rangle q' = \langle b \rangle q''$. Since $A$ is non-spontaneous, $\text{supp}(q'') \subseteq \text{supp}(q) \cup \{b\}$. Let $c \in \text{supp}(q')$ and $c \neq a$; we have to show $c \in \text{supp}(q)$. Now $q' = (ab) \cdot q''$, so $\text{supp}(q') = \langle ab \rangle \cdot \text{supp}(q'') \subseteq (ab) \cdot (\text{supp}(q) \cup \{b\})$. Since $(a)q' = \langle b \rangle q''$, we have $b \not= q'$, so $c \not\in \{a, b\}$; thus, $c \in (ab) \cdot \text{supp}(q) \cup \{a\}$, hence $c \in \text{supp}(q)$.

$A$ is name-dropping: Let $N \subseteq \text{supp}(q)$ for a state $q$, and let $q|_N$ restrict $q$ to $N$ in $A$; we show that $q|_N$ also restricts $q$ to $\bar{A}$. The support of $q|_N$ stays unchanged in $\bar{A}$, so we only have to check that $q|_N$ retains the requisite transitions. Throughout, it suffices to check new transitions.

For incoming transitions, let $p \xrightarrow{a} q$ in $\bar{A}$ be new, i.e. by Fact A.24 we have $a \in \text{supp}(p)$, $p \xrightarrow{b} q'$ in $A$, $b \not= p$ (hence $a \not= b$), and $\langle b \rangle q' = \langle a \rangle q$. Then $(ab) \cdot q = q'$. Therefore, $(ab) \cdot (q|_N)$ restricts $q'$ to $(ab) \cdot N$ in $A$ by Lemma A.23. It follows that $p \xrightarrow{ab} (ab) \cdot (q|_N)$ in $A$. Since $a \not= b$ and $(b)q' = \langle a \rangle q$, we have $a \not= q'$ and therefore $a \not= ((ab) \cdot (q|_N))$, so $\langle b \rangle ((ab) \cdot (q|_N)) = \langle a \rangle (q|_N)$ and therefore $p \xrightarrow{a} q|_N$ by construction of $\bar{A}$.

For outgoing transitions, let $q \xrightarrow{a} q'$ be new in $\bar{A}$; i.e. by Fact A.24 we have $a \in \text{supp}(q)$, $q \xrightarrow{b} q''$ in $A$, $b \not= q$ (hence $a \not= b$) and $\langle b \rangle q'' = \langle a \rangle q'$. Since $a \in \text{supp}(q)$, we have to show that $q|_N \xrightarrow{a} q'$ in $\bar{A}$, assuming $\text{supp}(q|_N) \cup \{a\} \subseteq N$. From $b \not= q$ we have $b \not= q|_N$, so by construction of $\bar{A}$, it suffices to show $q|_N \xrightarrow{b} q''$ in $A$, which will follow once we show $\text{supp}(q'') \subseteq N \cup \{b\}$. So let $b \not= c \in \text{supp}(q'')$; we have to show $b \in N$. Now $a \not= b$ and $(b)q'' = \langle a \rangle q'$ imply $a \not= q''$, so $c \not= a$ and hence $c \not\in \{a, b\}$. Therefore $c \in (ab) \cdot \text{supp}(q'') = \langle ab \rangle \cdot q'' = \text{supp}(q) \subseteq N$, as required.

$A$ is equivalent to $A$: $L(A) \subseteq L(\bar{A})$ is immediate as $A \subseteq \bar{A}$ by construction. For the reverse inclusion, we show that

\((*)\) whenever $w \in L(\bar{A}, q)$ then there exists $N \subseteq \text{supp}(q)$ such that if $q|_N$ restricts $q$ to $N$ in $A$ then $w \in L(A, q|_N)$

(in fact, $N$ will be such that $|\text{supp}(q) - N| \leq 1$). Since $\text{supp}(s) = \emptyset$ for the initial state $s$, this implies that $L(\bar{A}) \subseteq L(A)$.

We prove $(*)$ by induction on $w$, with trivial induction base. So let $w = av$ and $q \xrightarrow{a} q'$ in $\bar{A}$ where $v \in L(\bar{A}, q')$. By induction, there is $N \subseteq \text{supp}(q')$ such that $v \in L(A, q'|_N)$ whenever $q'|_N$ restricts $q'$ to $N$ in $A$. If $q \xrightarrow{a} q'$ in $A$ then $q|_N \xrightarrow{a} q'|_N$, so that $av \in L(A, q)$, the remaining case is that $q \xrightarrow{a} q'$ is new. By Fact A.24, we have $a \in \text{supp}(q)$ and $b, q''$ such that $q \xrightarrow{b} q''$ in $A$, $b \not= q$ (so $a \not= b$), and $\langle b \rangle q'' = \langle a \rangle q'$. We claim that whenever $q|_{N_\alpha}$ restricts $q$ to $N_\alpha := \text{supp}(q) - \{a\}$ in $A$ then $av \in L(A, q|_{N_\alpha})$. It suffices to show

$q|_{N_\alpha} \xrightarrow{a} q'|_N$ in $A$.  \hspace{0.5cm} (14)

Since $a \not= b$ and $\langle a \rangle q' = \langle b \rangle q''$, we have $a \not= q''$ so from $q \xrightarrow{b} q''$ in $A$ we obtain $\text{supp}(q'') \subseteq \{b\} \cup N_\alpha$ by non-spontaneity of $A$. By the definition of restriction, it follows that $q|_{N_\alpha} \xrightarrow{b} q''$ in $A$ (recall that $b \not= q$). Since $a \not\in \text{supp}(q|_{N_\alpha}) = N_\alpha$, we obtain

\footnote{For greater clarity we write $L(A, q)$ for $L(q)$ where $q$ is a state in $A$.}
by equivariance of transitions that \( q|_{N_a} \overset{a}{\rightarrow} q' \), which implies (14) by the definition of restriction: we have \( q' = (ab) \cdot q'' \) which implies

\[
\text{supp}(q') = (ab) \cdot \text{supp}(q'') \subseteq \{a\} \cup (ab) \cdot N_a = \{a\} \cup N_a,
\]

where the last step holds since \( a, b \notin N_a \).

Proof (Additional proof details for Corollary 6.7). It is straightforward to verify that non-spontaneous name-dropping NOFAs are closed under the standard product construction; specifically, given a state \((q_1, q_2)\) in a product automaton and \( N \subseteq \text{supp}(q_1, q_2) = \text{supp}(q_1) \cup \text{supp}(q_2) \), one checks readily that if \( q_i|_{N_i} \) restricts \( q_i \) to \( N_i := N \cap \text{supp}(q_i) \) for \( i = 1, 2 \), then \( (q_1|_{N_1}, q_2|_{N_2}) \) restricts \( (q_1, q_2) \) to \( N \).

Details for Remark 6.8  We show that the data language

\[
L = \{w, v \in A^*, a \in A\}
\]

is not accepted by any DOFA. Assume for a contradiction that \( A \) is a DOFA that accepts \( L \). Let \( n \) be the maximal size of a support of a state in \( A \). Let \( w = a_1 \ldots a_{n+1} \) for distinct \( a_i \), and let \( q \) be the state reached by \( A \) after consuming \( w \). Then there is \( i \in \{1, \ldots, n+1\} \) such that \( a_i \notin \text{supp}(q) \). Pick a fresh name \( b \). Then \( \delta(a_i, q) \) is final and \( \delta(b, q) \) is not; but since \( \delta(a_i, q) = (a_i b) \cdot \delta(b, q) \), this is in contradiction to equivariance of the set of final states.

Details on Name-Dropping Register Automata. By definition, the full class of name-dropping RAs is characterized by the existence, for every configuration \((c, w)\) and subset \( N \) of the names appearing in \( w \), of a configuration \((c, w)|_N\) that restricts \((c, w)\) to \( N \). A priori, nothing excludes the possibility that \((c, w)|_N\) uses a location other than \( c \). We do not anticipate that any expressivity can be gained from that (we refrain from proving this formally as we currently wish to give lower rather than upper estimates for the expressivity of name-dropping RAs) and therefore concentrate on name-dropping RAs where restrictions use the same location; these are the forgetful RAs mentioned in Section 6. Since there is only one transition constraint between any two locations, forgetful RAs are given by a condition concerning the individual transition constraints \( c \overset{\phi}{\rightarrow} c' \); recall that \( \phi \) is an equivariant subset of \( R \times A \times R \) where \( R = (A \cup \{\bot\})^n \) is the set of register assignments. For a register assignment \( w = (w_1, \ldots, w_n) \) write \( |w| = \{w_1, \ldots, w_n\} \cap A \) and for \( N \subseteq A \) let \( w|_N \) be defined as in Section 6, i.e. by \( (w|_N)_i = w_i \) for \( w_i \in N \) and \( (w|_N)_i = \bot \) otherwise. Explicitly, an RA is forgetful iff every transition constraint \( c \overset{\phi}{\rightarrow} c' \) in it satisfies

1. Non-spontaneity: \((w, a, v) \in \phi \) implies \(|v| \subseteq \{a\} \cup |w|\).
2. Name-dropping: For all \((w, a, v) \in \phi \),
   (a) \((w, a, v|_N) \in \phi \) for all \( N \subseteq |v|\);
   (b) if \( \{a\} \cup |v| \subseteq N \subseteq |w| \) then \((w|_N, a, v) \in \phi \);
   (c) if \( a \notin |w| \supseteq N \) and \(|v| \subseteq \{a\} \cup N \) then \((w|_N, a, v) \in \phi \).
Constraints satisfying Conditions 2a–2c are clearly closed under unions and intersections. To see that positive Boolean combinations of the four types of basic constraints \( \text{cmp}, \; \text{store}, \; \text{fresh}, \; \text{keep} \) given in Section 6 satisfy these conditions, it thus suffices to show that the basic constraints satisfy them. This is immediate for Condition 2a as \( \text{store} \) and \( \text{keep} \) both allow \( v_1 = \perp \). Conditions 2b and 2c are immediate for \( \text{store} \) and \( \text{keep} \) (in the case of \( \text{keep} \), again because it allows \( v_1 = \perp \)). To see that they hold for \((w, a, v) \in \text{fresh}\), just note that \((w|_N)_i \in \{w_i, \perp\}\) and \(a \neq \perp\). Condition 2c does not apply to \((w, a, v) \in \text{cmp}\). Finally, Condition 2b holds for \((w, a, v) \in \text{cmp}\) because \(a \in N\) and \(a = w_i\) imply that \((w|_N)_i = w_i = a\).

**Translation of FSUBAs into RNNAs.** We now compare RNNAs to finite-state unification-based automata (FSUBAs) [19,35]. A particular feature of FSUBAs is that they distinguish a finite subset \( \Theta \) of the alphabet that is read-only, i.e. cannot be written into the registers. We have no corresponding feature, therefore restrict to \( \Theta = \emptyset \) in the following discussion. An FSUBA then consists of finite sets \( Q \) and \( r \) of states and registers, respectively, a transition relation \( \mu \subseteq Q \times r \times \mathcal{P}_r(r) \times Q \), an initial state \( q_0 \in Q \), a set \( F \subseteq Q \) of final states, and an initial register assignment \( u \). Register assignments are partial maps \( v : r \to \mathbb{A} \), which means a register \( k \in r \) can be empty \( (v(k) = \perp) \) or hold a name from \( \mathbb{A} \). An FSUBA configuration is a pair \((q, v)\), where \( q \in Q \) and \( v \) is a register assignment. The initial configuration is \((q_0, u)\). A transition \((q, k, S, p) \in \mu\) applies to a configuration with state \( q \) for an input symbol \( a \in \mathbb{A} \) if register \( k \) is empty or holds \( a \); the resulting configuration has state \( p \), with the input \( a \) first written into register \( k \) and the register contents from \( S \) cleared afterwards. A word is accepted if there is a sequence of transitions from \((q_0, u)\) to a configuration with a final state.

As the name unification-based suggests, FSUBAs can check equality of input symbols, but not inequality (except with respect to the read-only letters); in other words, they have no notion of freshness. Thus the above-mentioned language \( \{aba \mid a \neq b\} \) cannot be accepted by an FSUBA [19].

We proceed to show that for every FSUBA \( A \) with empty read-only alphabet its configurations form an RNN that accepts the same \( \mathbb{A} \)-language as \( A \) under local freshness semantics; that is, RNNAs are strictly more expressive than FSUBAs with empty read-only alphabet.

Let \( A \) be an FSUBA with set \( Q \) of state, set \( r \) of registers, initial state \( q_0 \), set \( F \) of final states, transition relation \( \mu \subseteq Q \times r \times \mathcal{P}_r(r) \times Q \), and initial register assignment \( u \); we restrict the read-only alphabet \( \Theta \) to be empty. We denote the \( \mathbb{A} \)-language accepted by \( A \) by \( L(A) \). We construct an equivalent RNN \( R(A) \) as follows. The states of \( R(A) \) are the configurations of \( A \), which form a nominal set \( C \) under the group action \( \pi \cdot (q, v) = (q, \pi \cdot v) \). The transitions of \( R(A) \) are given by

\[
\text{fsuc}(q, v) = \{(v(k), (p, \text{erase}_S(v)) \mid (q, k, S, p) \in \mu\} \\
\cup \{(a, (p, \text{erase}_S(v[k \mapsto a])\mid (q, k, S, p) \in \mu, a \in \text{supp}(v), v(k) = \perp)\}
\]
\[
\text{bsuc}(q, v) = \{(a) (p, \text{erase}_S(v[k \mapsto a])) \mid (q, k, S, p) \in \mu, a \neq v, v(k) = \perp\}
\]

where \( \text{erase}_S \) clears the contents of the registers in \( S \).

This RNN \( R(A) \) behaves, under local freshness semantics, like the FSUBA \( A \):
Lemma A.25. The transitions between configurations of A are precisely given by $(q, v) \xrightarrow{ub(\alpha)} (p, w)$, where $(q, v) \xrightarrow{\alpha} (p, w), \alpha \in \mathbb{A}$, is a transition in $R(A)$.

Proof. Let $(q, v) \xrightarrow{\alpha} (p, w)$ be a transition in the RNNA $R(A)$. We distinguish cases:

- For (15), we have an FSUBA transition $(q, k, S, p) \in \mu$ with $\alpha = v(k) \in \mathbb{A}$, and $w = \text{erase}_S(v)$. Hence we have a transition $(q, v) \xrightarrow{ub(\alpha)} (p, w)$ between FSUBA configurations.

- For (16), we have an FSUBA transition $(q, k, S, p) \in \mu$ and $v(k) = \bot$, $\alpha = v(i) \in \mathbb{A}$ for some $i \in r$, and $w = \text{erase}_S(v[k \mapsto v(i)])$. Hence, from the FSUBA configuration $(q, v)$ the input $v(i)$ is read into register $k$ and then the registers in $S$ are cleared, i.e. $(q, v) \xrightarrow{\alpha} (p, w)$ is a transition of FSUBA configurations.

- For (17), i.e. for $\alpha = 1a$, we have an FSUBA transition $(q, k, S, p') \in \mu$ and $v(k) = \bot$ and some $b \not\# v$ with $\langle a \rangle(p, w) = \langle b \rangle(p', \text{erase}_S(v[k \mapsto b]))$. It follows that $(ab)(p, w) = (p', \text{erase}_S(v[k \mapsto b]))$, and equivalently, $p = p'$ and $(ab)w = \text{erase}_S(v[k \mapsto b])$. The latter implies that $w = \text{erase}_S(v[k \mapsto a])$. Thus, we obtain a transition of FSUBA configurations $(q, v) \xrightarrow{\alpha} (p, w)$ as desired.

Conversely, consider a transition $(q, v) \xrightarrow{\alpha} (p, w)$ of FSUBA configurations admitted by $(q, k, S, p) \in \mu$.

- If $v(k) \neq \bot$, then $v(k) = a$. Hence $(q, v) \xrightarrow{\alpha} (p, w)$ is a transition in $R(A)$ by (15).

- If $v(k) = \bot$ and $a \in \text{supp}(v)$, then $(q, v) \xrightarrow{\alpha} (p, w)$ is a transition in $R(A)$ by (16).

- If $v(k) = \bot$ and $a \not\# v$, then $w = \text{erase}_S(v[k \mapsto a])$ and $\langle a \rangle(p, w) \in \text{bsuc}(q, v)$.

By $\alpha$-invariance, this implies $(q, v) \xrightarrow{1a} (p, w)$ in $R(A)$.

Using Lemma A.25, one shows by induction on $w$ that $L(A) = \{ub(w) \mid w \in L_0(R(A))\}$. The RNNA $R(A)$ in general fails to be name-dropping, but for any $[lw]_{\alpha} \in L_{\alpha}(q, v), w \in \mathbb{A}^*$, we have

$$(q, v) \xrightarrow{lw} (p, w'), w \in L_{\alpha}(p, v') \quad \text{or} \quad (q, v) \xrightarrow{lw} (p, v'), w \in L_{\alpha}(p, v') : \quad (18)$$

Since $[lw]_{\alpha} \in L_{\alpha}(q, v), w$, we have some transition $(q, v) \xrightarrow{lw} (p', v'')$ in $R(A)$ such that $\langle a \rangle w = \langle b \rangle w'$ for some $v' \in L_{\alpha}(p', v'')$; if we cannot $\alpha$-equivalently rename the $lw$-transition into an $la$-transition to obtain the left alternative in (18), then $b \neq a \in \text{supp}(v'')$ and hence $a \in \text{supp}(v)$, so by construction of $R(A)$ we obtain the right alternative in (18). By induction on $w$, it follows that $\{ub(w) \mid w \in L_0(q_0, u)\} = D(L_0(q_0, u))$, so that $L(A) = D(L_0(R(A)))$, as claimed.

Relationship to Unambiguous Register Automata We have seen that the languages accepted by RNNA under the local freshness semantics RNNA are a proper subclass of languages accepted by RAs. An important subclass of RAs are the unambiguous ones. An RA $A$ is called unambiguous if every input word has at most one accepting run in $A$. The class of languages accepted by unambiguous RA lies strictly between those accepted by deterministic and by arbitrary (non-deterministic) RAs. Unambiguous RA are closed under complement and have decidable universality, language containment,
and equivalence problems. Moreover, a language is accepted by an unambiguous RA iff both itself and its complement are accepted by an RA (these results were announced in [9, Theorems 12 and 13].

It follows that unambiguous RA are incomparable to RNNA under local freshness semantics. Indeed, as we mentioned in the introduction, the language

\[ L = \{ ab \mid a, b \in A, a \neq b \} \]  

(19)

is not accepted by any RNNA but can clearly be accepted by an unambiguous (even deterministic) RA with only one register.

Now consider the language \( L_2 \) of all words in which some letter occurs at least twice, which is defined by the bar expression

\[ (a)^*a(b)^*a \]

and is therefore accepted by an RNNA (hence also by an RA). Then \( L_2 \) is not accepted by an unambiguous RA. For if it were then the complement of \( L_2 \),

\[ \{ a_1 \cdots a_n \mid n \in \mathbb{N}, a_i \text{ pairwise distinct} \} \]

(20)

would be accepted by an RA (by [9, Theorem 13]), which is known to be impossible [5].

**Relationship to Alternating 1-Register Automata**

Another model with decidable language inclusion are alternating 1-register automata (see e.g. [29]). We shall prove that they are incomparable to RNNA under local freshness semantics. For the failure of the inclusion of alternating 1-register automata into RNNA, just recall that the language \( L \) defined in Equation (19), which is not accepted by any RNNA, can be accepted by an alternating, even deterministic, 1-register automaton.

Alternating 1-register automata are fairly expressive, and in fact can express the language (20) as well as, maybe unexpectedly, the language ‘the first and the second letter both appear again, not necessarily in the same order’. The latter is due to the fact that the place where the automaton needs to look for the second occurrence of the second letter does not depend on where it finds the second occurrence of the first letter, so it can run independent searches for the second occurrences of the respective letters in two conjunctive branches, each using only one register.

However, we shall show that the language that is given under local freshness semantics by the regular bar expression

\[ |[a][b][c]^*ab[c]^*|, \]

and can therefore be accepted by an RNNA, cannot be accepted by any alternating 1-register automaton.

We first recall the definition of the latter as presented in [10]. We restrict to the case where the finite component of the input alphabet (just called the alphabet in [10]) is a singleton, matching our example.

**Definition A.26.** An alternating oneway 1-register automaton (A1-RA) consists of

1. a finite set \( Q \) of locations with an initial location \( q_0 \in Q \);
2. an one-step transition function step : \( Q \to \Delta(Q) \) where

\[
\Delta(Q) = \{ p_1 \not\in \uparrow \uparrow p_2, p_1 \lor p_2, p_1 \land p_2, \top, \bot, \uparrow p, \downarrow p, \exists p, \forall p \mid p, p_1, p_2 \in Q \};
\]

3. a height function \( \gamma : Q \to \mathbb{N} \) such that whenever \( \operatorname{step}(q) \in \{ p_1 \not\in \uparrow \uparrow p_2, p_1 \lor p_2, p_1 \land p_2, \top, \bot, \uparrow p, \downarrow p \} \) we have \( \gamma(p), \gamma(p_1), \gamma(p_2) < \gamma(q) \).

**Notation A.27.** For any set \( A \subseteq A \) and any A1-RA we write

\[
Q_A = Q \times (A + \{\ast\})
\]

for the set of configurations \((q, a)\) where \( c \) is a location and \( a \) indicates content of the register, which is either empty or contains a letter from \( A \). For the special case \( A = \mathbb{A} \), \( Q_A \) is a nominal set, with the nominal structure determined by taking \( Q \) to be discrete. We write \( i_A : Q_A \to Q_A \) for the inclusion map, and define \( u_A : Q_A \to Q_A \) by

\[
u_A(q, a) = \begin{cases} (q, a) & a \in A, \\ (q, \ast) & \text{otherwise.} \end{cases}
\]

(Of course, \( i_A \circ u_A : Q_A \to Q_A \) in general fails to be equivariant.)

**Remark A.28.** For \( A \subseteq B \subseteq \mathbb{A} \) we have

\[
i_A = i_B \cdot u_B \cdot i_A \quad \text{and} \quad u_A = u_A \cdot i_B \cdot u_B.
\] (21)

Input words from \( \mathbb{A}^* \) are provided on a read-only tape whose head may move to the right or stay in its position in every computation step. The automaton starts to run in the initial configuration \((q_0, \ast)\). For an input symbol \( i \in \mathbb{A} \), the behaviour of a configuration \((q, r) \in Q_A \) is determined by \( \operatorname{step}(q) \) as follows:

- For \( p_1 \lor p_2 \) (resp. \( p_1 \land p_2 \)), branch disjunctively (resp. conjunctively) into \( p_1 \) and \( p_2 \) without moving the head and changing the register content.
- For \( p_1 \not\in \uparrow \uparrow p_2 \), transfer to location \( p_1 \) if of \( i = r \), and to \( p_2 \) otherwise.
- For \( \top \) (resp. \( \bot \)), accept (resp. reject) instantly.
- For \( \downarrow p \), write \( i \) into the register and transfer to \( p \).
- For \( \exists p \) (resp. \( \forall p \)), transfer to \( p \) and move the head one step to the right to the next input symbol if there exists one; otherwise, reject (resp. accept).

It should be clear how the above determines acceptance of words by an A1-RA; see [10] for a more formal definition. We shall now explain how an A1-RA can be translated into an infinite deterministic automaton. To this end we consider the free Boolean algebra monad \( \mathcal{B} \) on \( \text{Set} \). Concretely, \( \mathcal{B} \) assigns to a set \( X \) the set of Boolean formulas built from elements of \( X \) using the binary operations \( \lor \) and \( \land \), the unary operation \( \neg \), and the constants \( \top, \bot \), modulo the axioms of Boolean algebras. We denote the unit of this monad by \( \eta : \text{Id} \to \mathcal{B} \). Each of its components \( \eta_X : X \to \mathcal{B}X \) is the universal map of the free Boolean algebra on \( X \), which embeds generators \( x \in X \) into \( \mathcal{B}X \). We note that \( \mathcal{B}Q_A \) is a nominal set, with the action of \( G \) given by \( \pi \cdot t = \mathcal{B}\pi(t) \) for \( t \in \mathcal{B}Q_A \) where we abuse \( \pi \) to denote the associated bijection \( Q_A \to Q_A \).

\textsuperscript{5} Branching on end-of-word as additionally foreseen in [10] can be encoded using the Boolean connectives and \( X \).
Definition A.29. Given an A1-RA as above, we define an function \( \delta : \mathbb{A} \times Q_A \rightarrow \mathcal{B}Q_A \) recursively by

\[
\delta(c, (q, r)) = \begin{cases} 
\delta(c, (p_1, r)) \circ \delta(c, (p_2, r)) & \text{if } \text{step}(q) = p_1 \circ p_2, \circ \in \{ \lor, \land \} \\
\delta(c, (p_1, r)) & \text{if } \text{step}(q) = p_1 \uparrow \uparrow p_2 \text{ and } c = r \\
\delta(c, (p_2, r)) & \text{if } \text{step}(q) = p_1 \downarrow \downarrow p_2 \text{ and } c \neq r \\
\text{step}(q) & \text{if } \text{step}(q) \in \{ \top, \bot \} \\
\delta(c, (p, c)) & \text{if } \text{step}(q) = \downarrow p \\
(p, r) & \text{if } \text{step}(q) = x p \\
-(p, r) & \text{if } \text{step}(q) = \bar{x} p
\end{cases}
\]

This recursion terminates because the height of states on the right-hand side is strictly smaller than \( \gamma(q) \).

The restriction of \( \delta \) to \( A \subseteq \mathbb{A} \) is defined by

\[
\delta_A = (\mathbb{A} \times Q_A \xrightarrow{\mathbb{A} \times i_A} \mathbb{A} \times Q_A \delta \xrightarrow{\mathcal{B}u_A} \mathcal{B}Q_A) \rightarrow \mathcal{B}Q_A.
\]

Note that \( \delta_A = \delta \).

It is easy to see that \( \delta \) is an equivariant function.

We now consider the curried version \( t_A : Q_A \rightarrow (\mathcal{B}Q_A)^{\mathbb{A}} \) of \( \delta_A \) and pair it with the constant map \( o_A = \bot! : Q_A \rightarrow 2 \), where \( 2 = \{ \bot, \top \} \). Since \( 2 \times (\mathcal{B}Q_A)^{\mathbb{A}} \) carries the obvious componentwise structure of a Boolean algebra, we can uniquely extend \( \langle o_A, t_A \rangle : Q_A \rightarrow 2 \times (\mathcal{B}Q_A)^{\mathbb{A}} \) to a Boolean algebra morphism

\[
\mathcal{B}Q_A \rightarrow 2 \times (\mathcal{B}Q_A)^{\mathbb{A}}. \tag{22}
\]

We write \( \text{acc}_A : \mathcal{B}Q_A \rightarrow 2 \) for the left-hand component and \( \delta^A_A : \mathbb{A} \times \mathcal{B}Q_A \rightarrow BQ_A \) for the currying of the right-hand one. As announced, \( \text{acc}_A \) and \( \delta_A \) determine the final states and the next state function of a deterministic automaton on \( \mathcal{B}Q_A \).

Remark A.30. The formation of (22) from \( \langle o_A, t_A \rangle \) is an instance of the generalized powerset construction [33]. Indeed, \( (\mathcal{B}, \langle o_A, t_A \rangle) \) is a coalgebra for the functor \( FT \) on \( \text{Set} \), where \( F = 2 \times (-)^{\mathbb{A}} \) is the type functor of deterministic automata considered as coalgebras and \( T = \mathcal{B} \) is the free boolean algebra monad. The functor \( F \) clearly lifts to the Eilenberg-Moore category of \( \mathcal{B} \) (i.e. the category of boolean algebras). Therefore, any coalgebra \( X \rightarrow FTX \) uniquely extends to the coalgebra \( TX \rightarrow FTX \) for the lifting of \( F \) to boolean algebras, viz. a deterministic automaton on the state set \( TX \).

As usual we extend any \( \delta^A_A \) to words by induction, which yields

\[
\delta^*_A : \mathbb{A}^* \times \mathcal{B}Q_A \rightarrow \mathcal{B}Q_A.
\]

From now on we shall abuse notation further and denote by \( \delta_A, \delta^A_A \) and \( \delta^*_A \) also their curried versions with codomain \( \mathcal{B}Q_A^{\mathbb{A}} \).

It is now straightforward to work out the following lemma (note that A1-RA as defined in [10] do not handle the empty word at all):
Lemma A.31. A given A1-RA accepts a non-empty word \( w \in \mathbb{A}^+ \) iff
\[
\text{acc}_A \cdot \delta^*_A(w) \cdot \eta(q_0, *) = \top.
\]

Lemma A.32. For any \( c, d \not\in A \), \( \delta_A(c) = \delta_A(d) \).

Proof. Since \( c, d \not\in A \), we have \( u_A = u_A \circ \pi \) where \( \pi : Q_A \to Q_A \) denotes the bijection associated to the transposition \((c \ d)\). Recall that the action of \( G \) on \( BQ_A \) yields \((c \ d) \cdot t = B\pi(t)\). Thus, for any \( (q, r) \in Q_A \), we obtain
\[
\begin{align*}
\delta_A(c, (q, r)) &= B(u_A(\delta(c, (q, r)))) \\
&= B(u_A B\pi(\delta(c, (q, r)))) \\
&= B(u_A(\delta((c \ d) \cdot (c, (q, r)))) \quad (\delta \text{ equivariant}) \\
&= B(u_A(\delta(d, (q, r)))) \\
&= \delta_A(d, (q, r)).
\end{align*}
\]

Corollary A.33. Let \( w, v \in \mathbb{A}^n \) be words that differ only in letters that appear only once in them, i.e. for all \( k < n \) if \( w_k \) (resp. \( v_k \)) appears again in \( w \) (resp. \( v \)) then \( w_k = v_k \). Let \( A \subseteq \mathbb{A} \) be the letters that occur more than once in \( w \) and \( v \). Then we have \( \delta^*_A(w) = \delta^*_A(v) \).

Proof. This follows from Lemma A.32 by induction on \( n \).

This means one can directly discard all those atoms of a word that appear only once and so we only need to keep those atoms in the register that appear at least twice.

Before coming to the main result, we show that we preserve the acceptance when restricting the register contents to those letters appearing twice in a word.

Lemma A.34. For all \( A \subseteq B \subseteq \mathbb{A}, c \in A \cup (\mathbb{A} \setminus B) \), the diagram
\[
\begin{array}{ccc}
BQ_B & \xrightarrow{\delta^*_B(c)} & BQ_B \\
\downarrow{B(u_A \cdot i_B)} & & \downarrow{B(u_A \cdot i_B)} \\
BQ_A & \xrightarrow{\delta^*_A(c)} & BQ_A
\end{array}
\]
commutes.

Proof. By universality of \( \eta_X : X \to BX \), it suffices to show that
\[
B(u_A \cdot i_B) \cdot \delta_B(c, (q, r)) = \delta_A(c) \cdot u_A \cdot i_B(q, r)
\]
for all \( q \in Q, r \in B + \{\star\} \). By the definition of \( \delta_A, \delta_B \) and Equation (21), this reduces to the equation
\[
B(u_A)(\delta(c, (q, r))) = B(u_A)(\delta(c, u_A(q, r))).
\]
This is trivial in case \( r \in A + \{\star\} \), so assume from now on that \( r \in B \setminus A \); it then remains to show that
\[
B(u_A)(\delta(c, (q, r))) = B(u_A)(\delta(c, (q, \star))). \tag{23}
\]
Intuitively, this means that the automaton model does not foresee branching on emptyness of the register. We prove (23) by induction on the height \( \gamma(q) \), distinguishing cases on the form of \( \text{step}(q) \). The Boolean cases are trivial. The remaining cases are as follows.

- For \( \text{step}(q) = p_1 \xleftarrow{\downarrow} p_2 \), note that our assumptions \( c \in A \cup (A \setminus B) \) and \( r \in B \setminus A \) imply that \( c \neq r \), so we have
  \[
  Bu_A(\delta_B(c, (q, r))) = Bu_A(\delta_B(c, (p_2, r))) = Bu_A(\delta_B(c, (p_2, \star))) \tag{IH}
  = Bu_A(\delta_B(c, (q, \star))).
  \]

- For \( \text{step}(q) = \downarrow p \), we have
  \[
  Bu_A(\delta(c, (q, r))) = Bu_A(\delta(c, (p, c))) = Bu_A(\delta(c, (p, \star))) \tag{IH}
  = Bu_A(\delta(c, (q, \star))).
  \]

- For \( \text{step}(q) = X p \), we have (using \( r \notin A \))
  \[
  Bu_A(\delta(c, (q, r))) = Bu_A(p, r) = (p, \star) = Bu_A(p, \star) = Bu_A(\delta(c, (q, \star))).
  \]

The case for \( \text{step}(q) = \bar{X} p \) is similar. \( \square \)

**Lemma A.35.** For \( w \in A^* \), \( B \supseteq \text{supp}(w) \), the diagram

\[
\begin{array}{ccc}
\mathcal{B}Q_B & \overset{\delta^B(w)}{\rightarrow} & \mathcal{B}Q_B \\
\downarrow Bi_B & & \downarrow Bi_B \\
\mathcal{B}Q_A & \overset{\delta^A(w)}{\rightarrow} & \mathcal{B}Q_A
\end{array}
\]

commutes.

**Proof.** Let \( c \in B \), \( q \in Q \), and \( r \in B + \{\star\} \). Since \( \delta_A \) is equivariant, we have \( \text{supp}(\delta_A(c, (q, r))) \subseteq B \), i.e. as a Boolean algebra term over variables from \( Q_A \), \( \delta_A(c, (q, r)) \) depends only on the variables from \( Q_B \). Therefore,
\[
\delta_A(c, (q, r)) = Bi_B \cdot Bu_B \cdot \delta_A(c, (q, r)). \tag{24}
\]
This implies that
\[
\delta^A_A(c) \cdot Bi_B \cdot \eta_Q = \delta_A(c) \cdot i_B
\]
\[
\overset{(24)}{=} Bi_B \cdot Bu_B \cdot \delta_A(c) \cdot i_B
\]
\[
= Bi_B \cdot \delta_B(c)
\]
\[
= Bi_B \cdot \delta^B_B(c) \cdot \eta_Q.
\]
Using the universal property of $\eta_{QB}$, we conclude that $\delta_A^\dagger(c) \cdot Bi_B = Bi_B \cdot \delta_B^\dagger(c)$, from which the main claim follows by induction on $w$.

In the following, we see that for certain words $w$, $\delta_A^\ast(w)$ behaves similarly to $\delta_A^\ast(c)$ for atoms $c \in \mathcal{A}$.

**Lemma A.36.** If $A \subseteq \mathcal{A}$ and $w \in \mathcal{A}^*$ contains only letters in $A$ more than once,

\[
\begin{array}{ccc}
B_{Q_A} & \xrightarrow{\delta_A^\ast(w)} & B_{Q_A} \\
\downarrow_{Bi_A} & & \downarrow_{Bi_A} \\
B_{Q_A} & \xrightarrow{\delta_A^\ast(c)} & B_{Q_A}
\end{array}
\]

then the diagram commutes.

**Proof.** Induction over the word length. For $w = \varepsilon$ the claim is clear because $\delta_A^\ast(\varepsilon)$ is just the identity and because $u_A \cdot i_A = id_{Q_A}$. For a word $wc$, $c \in \mathcal{A}$, $w \in \mathcal{A}^*$, $W = \text{supp}(w)$, $c \notin W \setminus A$ we have the commutativity of the diagram below:

\[
\begin{array}{ccc}
B_{Q_A} & \xrightarrow{\delta_A^\ast(w)} & B_{Q_A} \\
\downarrow_{Bi_A} & & \downarrow_{Bi_A} \\
B_{Q_A} & \xrightarrow{\delta_A^\ast(c)} & B_{Q_A} \\
\end{array}
\]

\[
\begin{array}{ccc}
B_{Q_A} & \xrightarrow{\delta_A^\ast(w)} & B_{Q_A} \\
\downarrow_{Bi_A} & & \downarrow_{Bi_A} \\
B_{Q_A} & \xrightarrow{\delta_A^\ast(c)} & B_{Q_A}
\end{array}
\]

**Lemma A.37.** If $A \subseteq B \subseteq \mathcal{A}$ and $w \in \mathcal{A}^*$ contains only letters in $A$ more than once, then

\[
\begin{array}{ccc}
B_{Q_B} & \xrightarrow{\delta_B^\ast(w)} & B_{Q_B} \\
\downarrow_{Bi_B} & & \downarrow_{Bi_B} \\
B_{Q_B} & \xrightarrow{\delta_B^\ast(c)} & B_{Q_B}
\end{array}
\]

\[
\begin{array}{ccc}
B_{Q_B} & \xrightarrow{\delta_B^\ast(w)} & B_{Q_B} \\
\downarrow_{Bi_B} & & \downarrow_{Bi_B} \\
B_{Q_B} & \xrightarrow{\delta_B^\ast(c)} & B_{Q_B}
\end{array}
\]

then the diagram commutes.

**Proof.** We have the following commutative diagram:
Proposition A.38. There is no A1-RA recognizing the language $ba|b|b^*ab|bd|^*$ under local freshness semantics, that is, the language of all words where the first two letters appear again later, in the same order and adjacent.

Proof. Assume for a contradiction that an A1-RA $(Q, q_0, \text{step})$ recognizes the language in question, and construct $\delta$ as described above. Pick any $a \neq b \in \mathbb{A}$. Let $n$ be greater than the (finite) cardinality of $B(Q_{\{a,b\}})$, and put $\ell = n!$. By assumption, the automaton accepts $abx$ with $x := \#^n ab \#^n$, keeping the concrete choice of $x$ fixed for the rest of the prove. We show that the automaton also accepts $aby$ with $y := \#^{n+1} b \#^{\ell-2} a \#^{n+1}$, which is clearly not an element of the language, again keeping $y$ fixed for the rest of the proof. For any $r \in \mathbb{A}$ we have that $\delta_r^* \left( \#^n \right) = \delta_r^* \left( \#^{n+\ell} \right)$: since $n$ is greater than $|B(Q_{\{r\}})|$, any run on $\#^n$ in $B(Q_{\{r\}})$ goes through a loop, and by Corollary A.33, the claim follows by iterating that loop, whose length divides $\ell$ by the choice of $\ell$. Since $a \neq b$, this implies that

\[
\begin{align*}
\delta_{\{a\}}^*(x) &= \delta_{\{a\}}^* \left( \#^n ab \#^n \right) = \delta_{\{b\}}^* \left( \#^n ab \#^n \right) \\
\delta_{\{b\}}^*(x) &= \delta_{\{b\}}^* \left( \#^n ab \#^n \right) = \delta_{\{b\}}^* \left( \#^n ab \#^{n+\ell} \right) \\
\delta_{\{a\}}^*(y) &= \delta_{\{a\}}^* \left( \#^{n+1} b \#^{\ell-2} a \#^{n+1} \right) = \delta_{\{b\}}^* \left( \#^{n+1} b \#^{\ell-2} a \#^{n+1} \right) = \delta_{\{b\}}^* (y). \quad (25)
\end{align*}
\]

For $r \in \{a, b\}$, the respective equality proves commutation of

\[
\begin{align*}
Q_{\{a,b\}} \xrightarrow{\eta Q_{\{a,b\}}} BQ_{\{a,b\}} &\xleftarrow{\delta^*_{\{a,b\}}(x)} BQ_{\{a,b\}} \\
Q_{\{r\}} \xrightarrow{\eta Q_{\{r\}}} BQ_{\{r\}} &\xleftarrow{\delta^*_{\{r\}}(x)} BQ_{\{r\}} \quad (25) \\
Q_{\{a,b\}} \xrightarrow{\eta Q_{\{a,b\}}} BQ_{\{a,b\}} &\xleftarrow{\delta^*_{\{a,b\}}(y)} BQ_{\{a,b\}}
\end{align*}
\]

Since $u_{\{a,b\}} \cdot i_{(r)} : Q_{\{r\}} \hookrightarrow Q_{\{a,b\}}$, $r = a, b$, are the inclusion maps, and since these are jointly surjective, we have

\[\text{acc}_{\{a,b\}} \cdot \delta^*_{\{a,b\}}(x) \cdot \eta Q_{\{a,b\}} = \text{acc}_{\{a,b\}} \cdot \delta^*_{\{a,b\}}(y) \cdot \eta Q_{\{a,b\}}.\]
and thus by the universality of $\eta_{Q_{\{a,b\}}}$ also
\[
\text{acc}_{\{a,b\}} \cdot \delta^*_\{a,b\}(x) = \text{acc}_{\{a,b\}} \cdot \delta^*_\{a,b\}(y)
\]  
(26)

Hence, $aby$ is accepted, because $abx$ is:

Relationship to 2-Register Automata Finally, we consider (nondeterministic) RAs with at most 2 registers, another class of automata with decidable language inclusion (see Kaminsky and Francez [18]). This class is also incomparable to RNNA. Indeed, the language (19) can be accepted even by a one-register RA but not by an RNNA. To see that the reverse inclusion also fails one considers the language ‘the first three letters appear again’. Clearly, this can be accepted by an RNNA, but not by any RA with at most 2 registers. Informally, such an RA would have to store the first three letters to compare each of them to their subsequent letters in the given input word; this is impossible with only 2 registers. A formal argument is similar to (but simpler than) the one given in Proposition A.38; we leave the details to the reader.

Note that essentially the same argument also shows RNNA to be incomparable to RA with at most $k$ registers for any fixed $k$; to see this consider the language ‘the first $k + 1$ letters appear again’.

A.5 Proofs and Lemmas for Section 7

Proof (Additional details for the proof of Theorem 7.1). We have omitted the space analysis of the initialization step. To initialize $\Xi$ we need to compute $N_2 = \text{supp}(L_\alpha(s_2))$. This can be done in nondeterministic logspace: for every free transition $q \overset{a}{\to} q'$ in $A_2$, in order to decide whether or not $a \in N_2$, remove from the transition graph of $A_2$ all transitions with label $a$ and then check whether there exists a path from $s_2$ to a final state passing through the given transition.

Details for Remark 7.2 The spines of an NKA expression $r$ arise by $\alpha$-renaming and subsequent deletion of some binders from expressions that consist of subexpressions of $r$, prefixed by at most as many binders as occur already in $r$; therefore, the degree of the RNNA formed by the spines, and hence, by Theorem 5.13 (and the fact that the translation from bar NFA to regular bar expressions is polynomial and preserves the degree), that of the arising regular bar expression, is linear in the degree of $r$ (specifically, at most twice as large).

We shortly write $D(w) = D(L_\alpha(w)) = \{ub'(w') \mid w' \equiv_\alpha w\}$ for $w \in \bar{A}^*$. 
Lemma A.39. If $w \subseteq w'$ then $D(w) \subseteq D(w')$.

Proof. Induction over $w$, with trivial base case. The only non-trivial case in the induction step is that $w = av$ and $w' = |av'|$ where $v \subseteq v'$. All bar strings that are $\alpha$-equivalent to $w$ have the form $au$ where $v \equiv_\alpha u$; we have to show $ub(au) \in D(|av'|)$.

We have $ub(u) \in D(v)$, so $ub(u) \in D(v')$ by induction; that is, there exists $v' \equiv_\alpha v'$ such that $ub(v') = ub(u)$. Then $ub(|av'|) = ub(|av|)$ and $|av'| \equiv_\alpha |av|$, so $au \in D(|av|)$.

Lemma 7.3 is immediate from the following:

Lemma A.40. Let $L$ be a regular bar language, and let $w \in \tilde{\kappa}^*$. Then $D(w) \subseteq D(L)$ iff there exists $w' \supseteq w$ such that $[w'_\alpha] \in L$.

Proof. ‘If’: If $[w'_\alpha] \in L$ then $D(w') \subseteq D(L)$, so $D(w) \subseteq D(L)$ by Lemma A.39.

‘Only if’: We generalize the claim to state that whenever

$$D(w) \subseteq \bigcup_{i \in I} D(L_\alpha(q_i))$$

for states $q_i$ in a name-dropping RNNA $A$ and a finite index set $I$, then there exist $i$ and $w' \supseteq w$ such that $[w'_{\alpha}] \in L_\alpha(q_i)$.

We prove the generalized claim by induction over $w$. The base case is trivial.

Induction step for words $aw$: Let $D(aw) \subseteq \bigcup_{i=1}^n D(L_\alpha(q_i))$. We prove below that

$$D(w) \subseteq \bigcup_{i \in I, q_i \xrightarrow{\alpha} q', \alpha \in \{a, \bar{a}\}} D(L_\alpha(q')).$$  \hspace{1cm} (27)

Indeed, let $u \in D(w)$, i.e. there exists $v \equiv_\alpha w$ with $ub(v) = u$. Then $ub(aw) = aw$ and $av \equiv_\alpha aw$ imply $au \in D(aw)$, so by assumption there exists $i \in \{1, \ldots, n\}$ such that $au \in D(L_\alpha(q_i))$, i.e. $au = ub(\alpha \bar{u})$ for $\alpha \in \{a, \bar{a}\}$ and $[\alpha \bar{u}]_{\alpha} \in L_\alpha(q_i)$. By Lemma 5.7, $\alpha \bar{u} \in L_0(q_i)$. Therefore there exists a transition $q \xrightarrow{\alpha \bar{u}} q'$ and $\bar{u} \in L_0(q')$. We conclude that $u = ub(\bar{u}) \in D(L_\alpha(q'))$ as desired.

Now, by induction hypothesis, it follows from (27) that we have $i \in I$, $\alpha \in \{a, \bar{a}\}$, $q_i \xrightarrow{\alpha} q'$, and $w' \supseteq w$ such that $[w'_{\alpha}] \in L_\alpha(q')$. Then $aw' \supseteq aw$ and $[aw'_{\alpha}] \in L_\alpha(q_i)$, as required.

Induction step for words $|aw$: Let $D(|aw|) \subseteq \bigcup_{i=1}^n D(L_\alpha(q_i))$. Notice that

$$D(|aw|) = \bigcup_{b=a \lor b \notin [w]_{\alpha}} bD(\pi_{ab} \cdot w)$$

(where $\cdot$ denotes the permutation group action and $\pi_{ab} = (a b)$ the transposition of $a$ and $b$; also note that $b \notin [w]_{\alpha}$ iff $b \notin \mathcal{F}N(w)$). Now pick $b \in A$ such that $b \notin [w]_{\alpha}$ and none of the $q_i$ has a $b$-transition (such a $b$ exists because the set of free transitions of each $q_i$ is finite, as $A$ is an RNNA). We prove below that

$$bD(\pi_{ab} \cdot w) \subseteq \bigcup_{i \in I, q_i \xrightarrow{\pi_{ab}} q'} bD(L_\alpha(q')),$$
and hence we have
\[ D(\pi_{ab} \cdot w) \subseteq \bigcup_{i \in I, q_i \xrightarrow{b} q'} D(L_{\alpha}(q')), \]  
(28)
again a finite union. In order to see that the above inclusion holds, let \( bu \in bD(\pi_{ab} \cdot w) \), i.e., we have \( v \equiv_{\alpha} \pi_{ab} \cdot w \) with \( ub(v) = u \). Then \( \|bv\|_{\alpha} = b(\pi_{ab} \cdot w) \equiv_{\alpha} \|aw\| \) and \( ub(\|bv\|) = bu \), which implies that \( bu \in D(\|aw\|) \). By our assumption \( D(\|aw\|) \subseteq \bigcup_{i=1}^{n} D(L_{\alpha}(q_i)) \) we obtain \( i \in \{1, \ldots, n\} \) such that \( bu \in D(L_{\alpha}(q_i)) \), i.e. \( bu = ub(\beta \bar{u}) \) for \( \beta \in \{b, \overline{b}\} \) and \( [\beta \bar{u}]_{\alpha} \in L_{\alpha}(q_i) \). By Lemma 5.7, we have \( \beta \bar{u} \in L_{0}(q_i) \), and since \( q_i \) has no \( b \)-transitions, we therefore know that \( \beta = b \). Hence we have a transition \( q_i \xrightarrow{b} q' \) and \( \bar{u} \in L_{0}(q') \). It follows that \( u = ub(\bar{u}) \in D(L_{\alpha}(q')) \), whence \( bu \in D(L_{\alpha}(q')) \) as desired.

Now, by induction hypothesis, we obtain from (28) \( i \in I, q_i \xrightarrow{b} q' \), and \( w' \equiv_{\pi_{ab}} w \) such that \( [w']_{\alpha} \in L_{\alpha}(q') \). It follows that
\[ \|bw'\| = b(\pi_{ab} \cdot w) \quad \text{and} \quad \|bw'\|_{\alpha} \in L_{\alpha}(q_i). \]
Now we have \( a \models [\pi_{ab} \cdot w]_{\alpha} \) (because \( b \models [w]_{\alpha} \)), and therefore \( a \models [w']_{\alpha} \) because \( \pi_{ab} \cdot w \subseteq w' \); it follows that \( la(\pi_{ab} \cdot w') \equiv_{\alpha} lw' \). As \( \sqsubseteq \) is clearly equivariant, we have \( \pi_{ab} \cdot w' \subseteq w \), so
\[ la(\pi_{ab} \cdot w') \sqsubseteq lw \quad \text{and} \quad [la(\pi_{ab} \cdot w')]_{\alpha} = [lw']_{\alpha} \in L_{\alpha}(q_i), \]
which proves the inductive claim.