Freshness and name-restriction in sets of traces with names

Murdoch J. Gabbay and Vincenzo Ciancia

Abstract. We use nominal sets (sets with names and binding) to define a framework for trace semantics with dynamic allocation of resources. Using novel constructions in nominal sets, including the technical devices of *positive nominal sets* and *maximal planes*, we define notions of capture-avoiding composition and name-restriction on sets of traces with names. We conclude with an extended version of Kleene algebras which summarises in axiomatic form the relevant properties of the constructions.

1 Introduction

Imagine a process evolving; every so often it may communicate with the outside world. In particular it may generate a new resource (allocate some memory; or perhaps create a cryptographic secret; or perhaps create a new channel name) and communicate that new resource.

Represent resources as *atoms* (set-theorists can think of *urelemente*; process-calculists can think of *names*). Then a model of the behaviour of our process is a trace (finite list of actions) that may contain atoms, and a model of the process is the set of all its possible traces. This is the *trace semantics* of computation [9], widely-used both in theory and application (e.g. the SPIN model-checker [17]).

Thus, a model of behaviour for processes with dynamic allocation, is sets of traces with atoms. But how then to represent dynamic allocation within this framework? After all, if a program outputs ab there is nothing in the string itself to tell us whether a, b, both a and b, or neither a nor b, has been 'created fresh'.

We propose to represent binding using a notion similar to that of nominal abstract syntax [15]; if e.g. ab is a possible trace of the process then the a in that trace is considered α-convertible when for all but finitely many a', $a'b$ is also in the set of possible traces of the process (note there is no binding primitive on the trace *itself*).

We will develop a model of binding based purely on sets of traces, such that:

- There is an operation νa that takes a behaviour (set of traces) X and creates another behaviour (also a set of traces) $\nu a.X$ in which a is α-renameable.
- Union of sets of traces is exactly union of sets.
- There is a notion of capture-avoiding composition $\circ$ such that $(\nu a.X) \circ \nu b.Y = \nu a.\nu b.(X \circ Y)$ if we choose a fresh for Y and b fresh for X.
  Thus $(\nu a.ab) \circ \nu b.b = \nu a.\nu b'.abb'$. Note that $ab \in \nu a.ab$ and $b \in \nu b.b$ but abb is *not* in their composition, because of capture-avoidance.

Finally we introduce axioms that regulate the behaviour of ν, by suggesting a notion of 'nominal' Kleene algebra [20].

We believe the constructions in this paper would be more generally applicable, so long as the semantics can be represented as sets and sensible notions of composition can be defined in some pointwise manner on their elements (in this paper it is sets of lists and list concatenation; see Definition 5.7).

Technical overview. The meat of this paper is some concrete calculations on nominal sets. The key technical facts are Theorems 3.14 and 3.16 and Proposition 4.6. The key definitions are Definitions 3.7, 4.5, and 5.7. The main theorem is Theorem 6.4.

Nominal sets were developed to represent syntax with binding; see [15] or a recent survey [13]. In this paper we use nominal sets to interpret sets of traces with atoms. The notions of names and free/bound names we use are exactly those from [15]; namely atoms and supporting set/freshness.

To the reader familiar with nominal sets, name-restriction $\nu a.X$ will be no surprise; Proposition 4.6 characterises it as a variation of atoms-abstraction $[a]x$ from [15] (see [13, Definition 3.8 and Lemma 3.13]). Readers familiar with presheaves will recognise this as a sets-based presentation of δ from e.g. [8] or [21]; see [16,7] for further discussion.

The difference, which is significant, is that X and $\nu a.X$ are of the same type (both are sets of traces); our name-restriction is not a monad in the sense of [22], though it does a similar job. More on this in the Conclusions.

Given that behaviour is modelled as 'just sets' and not wrapped up in an explicit abstraction, the challenge is that in composition $X \circ Y$, bound atoms in Y should somehow be detected and renamed to avoid capture with free atoms in X (see Remark 5.8), and vice-versa.

We use *maximal positive planes* as a foundational data structure for a notion of capture-avoiding language composition. Planes (Definition 3.3) are from [12] and were used to model capture-avoiding substitution. Positive planes are new (Definition 3.7), as is the connection with ν (Proposition 4.6). Arguably, planes and positive planes are as interesting as their application in this paper and we expect them to be useful in the future.

We conclude with an axiomatisation in the style of Kleene algebras and a proof of soundness (Section 6).

2 Nominal preliminaries

More on these constructions in [15] or in a recent survey [13].

Definition 2.1. Fix a countably infinite set $\mathbb{A}$ of **atoms**. We use a *permutative convention* that $a, b, c, \ldots$ range over *distinct* atoms.

A **permutation** π is a bijection on atoms such that $\mathit{nontriv}(\pi) = \{a \mid \pi(a) \neq a\}$ is finite. We write id for the **identity** permutation such that $\mathrm{id}(a) = a$ for all a. We write $\pi' \circ \pi$ for composition, so that $(\pi' \circ \pi)(a) = \pi'(\pi(a))$. We write π^{-1} for inverse, so that $\pi^{-1} \circ \pi = \mathrm{id} = \pi \circ \pi^{-1}$. Write $(a\ b)$ for the **swapping** permutation that maps a to b and b to a and all other c to themselves.

π will range over permutations.

Definition 2.2. If $A \subseteq \mathbb{A}$ define $fix(A) = \{\pi \mid \forall a \in A.\pi(a) = a\}$. A, B, C will range over finite sets of atoms.

Definition 2.3. A **set with a permutation action** is a pair $\mathsf{X} = (|\mathsf{X}|, \cdot)$ of an **underlying set** and a group action of permutations which we write $\pi{\cdot}x$ (so $\mathrm{id}{\cdot}x = x$ and $\pi'{\cdot}(\pi{\cdot}x) = (\pi' \circ \pi){\cdot}x$).

Say finite $A \subseteq \mathbb{A}$ **supports** $x \in |\mathsf{X}|$ when $\pi{\cdot}x = x$ for all $\pi \in fix(A)$. Say $x \in |\mathsf{X}|$ has **finite support** when it is supported by some finite A.

A **nominal set** is a set with a permutation action whose every element has finite support. X will range over nominal sets and x will range over elements $x \in |\mathsf{X}|$ (that is, finitely-supported elements).

Proposition 2.4. *If $A', A \subseteq \mathbb{A}$ are finite and support x then so does $A' \cap A$.*

Definition 2.5. Define $supp(x) = \bigcap\{A \subseteq \mathbb{A} \mid A \text{ finite and support } x\}$. Call $supp(x)$ the **support** of x.
Write $a\#x$ when $a \notin supp(x)$ and call a **fresh for** x.

We know $supp(x)$ is well-defined in Definition 2.5 because by assumption at the end of Definition 2.3, x has some finite supporting set.

Theorem 2.6. *$supp(x)$ supports x.*

Corollary 2.7.
1. *If $\pi(a) = a$ for all $a \in supp(x)$ then $\pi{\cdot}x = x$.*
2. *If $\pi(a) = \pi'(a)$ for every $a \in supp(x)$ then $\pi{\cdot}x = \pi'{\cdot}x$.*
3. *$a\#x$ if and only if $\exists b.b\#x \wedge (b\ a){\cdot}x = x$.*

Proposition 2.8. *$supp(\pi{\cdot}x) = \pi{\cdot}supp(x)$.*

Definition 2.9. Give $X \subseteq |\mathsf{X}|$ a **pointwise** permutation action given by $\pi{\cdot}X = \{\pi{\cdot}x \mid x \in X\}$.

Define $pow(\mathsf{X})$ to have elements $X \subseteq |\mathsf{X}|$ with the pointwise permutation action and finite support.

It is not hard to use Proposition 2.8 to verify that $pow(\mathsf{X})$ is a nominal set.

3 The planes of a set

From now until Section 6 we fix a nominal set X.

x, y, u, v will range over elements of $|\mathsf{X}|$ (i.e. $x, y, u, z \in |\mathsf{X}|$) and X, Y, Z, U, and V will range over finitely-supported subsets of $|\mathsf{X}|$ (i.e. $X, Y, Z, U, V \in |pow(\mathsf{X})|$).

3.1 Basic theory of planes

A subset $X \subseteq |\mathsf{X}|$ can be represented as a not-necessarily-disjoint union of orbits under certain subgroups of the permutation action. A canonical representation can be created using *planes* (Definition 3.3). A plane is an α-equivalence class

of an element under simultaneous renaming of one or more of its atoms (cf. Lemma 3.2). For more on planes see [12, Subsection 3.3].

Definition 3.1. Define the **plane** $x\!\!\leftsquigarrow_A$ by $x\!\!\leftsquigarrow_A = \{\pi{\cdot}x \mid \pi \in \mathit{fix}(A)\}$.
We may omit brackets, writing e.g. $x\!\!\leftsquigarrow_a$ for $x\!\!\leftsquigarrow_{\{a\}}$ and $x\!\!\leftsquigarrow_{a,b}$ for $x\!\!\leftsquigarrow_{\{a,b\}}$.

Lemma 3.2. *If* $\pi \in \mathit{fix}(A)$ *then* $(\pi{\cdot}x)\!\!\leftsquigarrow_A = x\!\!\leftsquigarrow_A$.

Lemma 3.2 expresses α-convertibility for (x, A) as a choice of representative for the plane $x\!\!\leftsquigarrow_A$, allowing us to rename $\mathit{supp}(x)\backslash A$. A plausible (if long) notation for $x\!\!\leftsquigarrow_A$ is $\nu(\mathit{supp}(x)\backslash A).\{x\}$ where ν is atoms-restriction (cf. Definition 4.5).

Definition 3.3. Suppose $A \subseteq \mathbb{A}$ is finite. Write $u\!\!\leftsquigarrow_A \propto Z$ when $u\!\!\leftsquigarrow_A \subseteq Z$ and for every u' and A', $u\!\!\leftsquigarrow_A \subseteq u'\!\!\leftsquigarrow_{A'} \subseteq Z$ implies $u'\!\!\leftsquigarrow_{A'} = u\!\!\leftsquigarrow_A$.
We call the plane $u\!\!\leftsquigarrow_A$ **maximal** in Z.

Example 3.4.
1. $a\!\!\leftsquigarrow_{\varnothing} \propto \mathbb{A}$ (this is the only maximal plane in $\mathbb{A}$).
2. $a\!\!\leftsquigarrow_{\{a\}} \propto \{a\} \cup (\mathbb{A} \times \mathbb{A})$, $(a,a)\!\!\leftsquigarrow_{\varnothing} \propto \{a\} \cup (\mathbb{A} \times \mathbb{A})$, and $(a,b)\!\!\leftsquigarrow_{\varnothing} \propto \{a\} \cup (\mathbb{A} \times \mathbb{A})$ (there are three maximal planes).
3. $a\!\!\leftsquigarrow_{\{b\}} \propto \mathbb{A} \backslash \{b\}$ (this is the only maximal plane in $\mathbb{A} \setminus \{b\}$).
4. In Definition 3.3, quantification over A' is necessary. For example: $a\!\!\leftsquigarrow_{\{a\}} \not\propto \mathbb{A} \setminus \{b\}$, but for no $A' \subsetneq \{a\}$ is $a\!\!\leftsquigarrow_{A'} \subseteq \mathbb{A} \setminus \{b\}$ (the only choice for A' is $\varnothing$).

Remark 3.5. For a fixed Z and u there may be two distinct subsets $A \subseteq \mathit{supp}(Z)$ and $B \subseteq \mathit{supp}(Z)$ such that $u\!\!\leftsquigarrow_A \propto Z$ and $u\!\!\leftsquigarrow_B \propto Z$ (thus; A and B are minimal but not *least*). For example, if $Z = \{(x,y) \mid x = a \vee y = b\}$ and $u = (a,b)$ then $u\!\!\leftsquigarrow_{\{a\}} \propto Z$ and $u\!\!\leftsquigarrow_{\{b\}} \propto Z$ but $u\!\!\leftsquigarrow_{\varnothing} \not\propto Z$.

Proposition 3.6. $Z = \bigcup\{u\!\!\leftsquigarrow_A \mid u\!\!\leftsquigarrow_A \propto Z\}$.

Proof. Planes in Z are ordered by subset inclusion. For each $x \in Z$, take some greatest element above $x\!\!\leftsquigarrow_{\mathit{supp}(x)}$. Their union is Z.

3.2 Positive planes

For the definition see Definition 3.7 and Example 3.8.

Planes do not have to be disjoint. This may be a surprise at first, since planes are orbits of an element under a permutation group and we are used to results stating that orbits are either equal or disjoint—but the groups could be different for different planes. For example, $(a,b)\!\!\leftsquigarrow_{\{a\}}$ and $(a,b)\!\!\leftsquigarrow_{\{b\}}$ are both positive planes, but $(a,b)\!\!\leftsquigarrow_{\{a\}} \cap (a,b)\!\!\leftsquigarrow_{\{b\}}$ is non-empty because it contains (a,b).

Theorem 3.16 expresses that a union of positive planes behaves as a coproduct in one respect: if a plane is included in a union of *positive* planes, then it is included in one of the planes that made up that union.

Thus, the interest of positive planes is that they need not be disjoint, but when we take their union it behaves in some ways like a coproduct. This fails if planes are not positive; see Corollary 3.18 and Example 3.19. The importance of this will become more apparent in the next section.

Definition 3.7. Recall the definition of $x\rangle_A$ from Definition 3.1 and the definition of $x\rangle_A \propto X$ from Definition 3.3.
Call $x\rangle_A$ **positive** when $A \subseteq supp(x)$.
Call X **positive** when every $x\rangle_A \propto X$ is positive.

Example 3.8. $\{a\} = a\rangle_{\{a\}}$ is positive. $\mathbb{A} \setminus \{a\} = b\rangle_{\{a\}}$ is not.

Lemma 3.9. *If $z\rangle_C \cap x\rangle_A \neq \varnothing$ and $A \subseteq C$ then $z\rangle_C \subseteq x\rangle_A$.*

Proof. Suppose $z' \in z\rangle_C \cap x\rangle_A$. Then $z' = \pi_z \cdot z = \pi_x \cdot x$ for some $\pi_z \in fix(C)$ and $\pi_x \in fix(A)$. Now $A \subseteq C$ so $fix(C) \subseteq fix(A)$. By Lemma 3.2 (twice) $z\rangle_C = z'\rangle_C = x\rangle_C \subseteq x\rangle_A$.

Proposition 3.10. *Suppose $u\rangle_A$ and $u'\rangle_{A'}$ are positive. Suppose $u \in u'\rangle_{A'}$.*
Then $u\rangle_A \subseteq u'\rangle_{A'}$ if and only if $A' \subseteq A$.

Proof. Suppose $u\rangle_A \subseteq u'\rangle_{A'}$ and $a \in A' \setminus A$. By assumption $A' \subseteq supp(u')$ so by Proposition 2.8, $a \in supp(v)$ for every $v \in u'\rangle_{A'}$. Also by assumption $A \subseteq supp(u)$ and so by Proposition 2.8, $a \notin supp((a'\ a) \cdot u)$ for fresh a' (so $a' \notin A' \cup supp(u)$). Now $(a'\ a) \in fix(A)$ so $(a'\ a) \cdot u \in u\rangle_A$, contradicting $u\rangle_A \subseteq u'\rangle_{A'}$.
Conversely, if $A' \subseteq A$ then we use Lemma 3.9.

Corollary 3.11. *Suppose $x\rangle_A \subseteq X$ and $x\rangle_A$ and X are positive. Then there is $A' \subseteq A$ with $x\rangle_A \subseteq x\rangle_{A'} \propto X$.*

Proof. Planes are ordered by subset inclusion, so there exist some x' and A' such that $x\rangle_A \subseteq x'\rangle_{A'} \propto X$. By assumption $x'\rangle_{A'}$ is positive. By Proposition 3.10 $A' \subseteq A$. We use Lemma 3.2 to conclude that $x'\rangle_{A'} = x\rangle_{A'}$.

Lemma 3.12. *If $x\rangle_A$ is positive then $supp(x\rangle_A) = A$.*

Proof. The left-to-right subset inclusion is from Lemma 3.2. Conversely suppose $a \in A$ and choose b fresh (so $b \notin A \cup supp(x)$). By positivity, $a \in supp(x)$. Using Proposition 2.8 it follows that $(b\ a) \cdot x \notin x\rangle_A$, so $(b\ a) \cdot (x\rangle_A) \neq x\rangle_A$. Thus $a \in supp(x\rangle_A)$.

Lemma 3.13. *If $u\rangle_A \propto Z$ and Z is positive then $A \subseteq supp(Z)$.*

Proof. Suppose $u\rangle_A \subseteq Z$ and $a \in A \setminus supp(Z)$. From Theorem 2.6 $(b\ a) \cdot (u\rangle_A) \subseteq Z$ for every $b \notin supp(Z)$; by group arguments $u\rangle_{A \setminus \{a\}} \subseteq Z$. Also $((b\ a) \cdot u)\rangle_{(b\ a) \cdot A} \propto Z$ for every $b \notin supp(Z)$. Since $u\rangle_A$ is positive, so is $u\rangle_{A \setminus \{a\}}$. By Lemma 3.12 and Proposition 3.10 $u\rangle_A \subsetneq u\rangle_{A \setminus \{a\}}$, contradicting maximality of $u\rangle_A$.

Theorem 3.14. *If Z is positive then $supp(Z) = \bigcup\{A \mid u\leftsquigarrow_A \propto Z\}$.*

Proof. The right-to-left inclusion is by Lemma 3.13. The left-to-right inclusion then follows from Proposition 3.6 and Lemma 3.12 using [13, Theorem 2.29].[1]

Remark 3.15. Theorem 3.14 fails if we remove the condition of maximality. For instance, $supp(\mathbb{A}) = \varnothing$ and $a\leftsquigarrow_{\{a\}}$ is non-maximal in $\mathbb{A}$ for every a, but $supp(a\leftsquigarrow_{\{a\}}) = \{a\}$ and $\bigcup\{supp(a\leftsquigarrow_{\{a\}} \mid a \in \mathbb{A}\} = \mathbb{A} \neq \varnothing$.

Theorem 3.16. *Suppose I is some indexing set and for each $i \in I$, A_i is a finite set of atoms and x_i is some element. Suppose $x_i\leftsquigarrow_{A_i}$ is positive for every $i \in I$ and $\bigcup A_i$ is finite. Then $z\leftsquigarrow_C \subseteq \bigcup x_i\leftsquigarrow_{A_i}$ implies $z\leftsquigarrow_C \subseteq x_i\leftsquigarrow_{A_i}$ for some i.*

Proof. We will show that $A_i \subseteq C$ for some $i \in I$ with $z \in x_i\leftsquigarrow_{A_i}$; the result follows by Proposition 3.10. Suppose otherwise, so that $\forall i.\, z{\in}x_i\leftsquigarrow_{A_i} \Rightarrow A_i{\not\subseteq}C$. Choose some π mapping $(\bigcup A_i) \setminus C$ to fresh atoms. For each i there are two possibilities: if $A_i \subseteq C$ then $z \notin x_i\leftsquigarrow_{A_i}$ and by Lemma 3.2 $\pi{\cdot}z \notin x_i\leftsquigarrow_{A_i}$; if $A_i \not\subseteq C$ then using Proposition 2.8 again $\pi{\cdot}z \notin x_i\leftsquigarrow_{A_i}$. Yet by construction $\pi \in fix(C)$ so $\pi{\cdot}z \in z\leftsquigarrow_C$, contradicting our assumption that $z\leftsquigarrow_C \subseteq \bigcup x_i\leftsquigarrow_{A_i}$.

Remark 3.17. Theorem 3.16 ensures that every maximal plane of a union of positive planes is one of the planes of that union.[2] We use this for example in Theorem 5.11 and Lemma 5.13.

Corollary 3.18. *Suppose $x_i\leftsquigarrow_{A_i}$ is positive for every $i \in I$, and $\bigcup A_i$ is finite. Then $z\leftsquigarrow_C \propto \bigcup x_i\leftsquigarrow_{A_i}$ if and only if $z\leftsquigarrow_C = x_i\leftsquigarrow_{A_i}$ for some i.*

Proof. By Theorem 3.16 $z\leftsquigarrow_C \subseteq x_i\leftsquigarrow_{A_i}$ for some i. We use maximality of $z\leftsquigarrow_C$.

Example 3.19. $U = (\mathbb{A}{\times}\mathbb{A}) \setminus \{(a,b)\}$ is not positive. $U \cup \{(a,b)\} = \mathbb{A} \times \mathbb{A}$; then $(a,b)\leftsquigarrow_\varnothing$ is maximal in $\mathbb{A} \times \mathbb{A}$ but is not a subset of U or $\{(a,b)\}$. Thus, it is not possible to retrieve from a union of not-necessarily-positive planes a subcollection of planes that make up that union. Extending this paper to the 'negative' case is future work.

Corollary 3.20. *X is positive if and only if $X = \bigcup x_i\leftsquigarrow_{A_i}$ for some set of positive $x_i\leftsquigarrow_{A_i}$ such that $\bigcup A_i$ is finite.*

Proof. If X is positive the result follows taking $x_i\leftsquigarrow_{A_i} \propto X$ and using Theorem 3.14. Conversely suppose $X = \bigcup x_i\leftsquigarrow_{A_i}$ for some set of positive $x_i\leftsquigarrow_{A_i}$ and suppose $\bigcup A_i$ is finite. We use Corollary 3.18.

---

[1] Positivity is sufficent but not necessary. It suffices to restrict to $u\leftsquigarrow_A$ such that $\neg(supp(u) \subsetneq A)$. This excludes an artefact of representing planes as pairs (u, A) since if $supp(u) \subsetneq A$ and $supp(u) \subsetneq B$ then $u\leftsquigarrow_A = u\leftsquigarrow_B = u\leftsquigarrow_{supp(u)}$. Since we concentrate on positive planes, where this cannot happen, we can ignore this.

[2] Contrast with $\mathbb{A} \setminus \{a\} = \bigcup\{b\leftsquigarrow_{\{b\}} \mid b \in \mathbb{A} \setminus \{a\}\} = b\leftsquigarrow_{\{a\}} \not\subseteq b\leftsquigarrow_{\{b\}}$. Theorem 3.16 is inapplicable because $\bigcup\{supp(b\leftsquigarrow_{\{b\}}) \mid b \in \mathbb{A} \setminus \{a\}\}$ is not finite.

4 ν-restriction on nominal sets

We are now ready to define a notion of name-restriction on (positive) sets (Definition 4.5). By Theorem 3.16 the planes of $\nu a.U$ are 'the planes of U, with a taken out of the support of each'. Proposition 4.6 relates that to a notion of 'U, with a taken out of the support of U' which resembles the nominal atoms-abstraction from [15] (see [13, Definition 3.8 and Lemma 3.13] for a proof). The rest of this section proves some useful equalities involving name-restriction.

Definition 4.1. Suppose X is a set. Define $X\frown_A$ and $X\frown^A$ by:

$$X\frown_A = \{\pi{\cdot}x \mid x \in X,\ \ \pi \in \mathit{fix}(A)\}$$
$$X\frown^A = \{\pi{\cdot}x \mid x \in X,\ \ \pi \in \mathit{fix}(\mathit{supp}(X) \setminus A)\}$$

Write $X\frown^a$ for $X\frown^{\{a\}}$. That is: $X\frown^a = \{\pi{\cdot}x \mid x{\in}X,\ \pi{\in}\mathit{fix}(\mathit{supp}(X){\setminus}\{a\})\}$.

Example 4.2. $(\mathbb{A}\setminus\{a\})\frown^a = \mathbb{A}$. For comparison, $(\mathbb{A}\setminus\{a\})\backsim_\varnothing = \{\mathbb{A}\setminus\{x\}|x{\in}\mathbb{A}\}$.

Lemma 4.3. *If $B \cap \mathit{supp}(X) = B' \cap \mathit{supp}(X)$ then $X\frown_B = X\frown_{B'}$.*

Lemma 4.4. 1. *Suppose $\mathit{supp}(u) \cap B \subseteq \mathit{supp}(u) \cap A$. Then $u\backsim_A\frown^B = u\backsim_{A\setminus B}$.*
2. *Suppose X is positive. Then $X\frown^B = \bigcup\{u\backsim_A\frown^B \mid u\backsim_A \propto X\}$.*

Proof. The first part is by routine arguments using part 2 of Corollary 2.7.
For the second part, using Proposition 3.6 we can calculate that

$$X\frown^B = \bigcup\{u\backsim_A\frown_{\mathit{supp}(X)\setminus B} \mid u\backsim_A \propto X\}.$$

By assumption each $u\backsim_A \propto X$ is positive. By Lemma 3.12 $\mathit{supp}(u\backsim_A) = A$. By Lemma 4.3 $u\backsim_A\frown_{\mathit{supp}(X)\setminus B} = u\backsim_A\frown_{A\setminus B}$ and by definition this is equal to $u\backsim_A\frown^B$.

Definition 4.5. Define $\nu a.U = \bigcup\{u\backsim_{A\setminus\{a\}} \mid u\backsim_A \propto U\}$.

Proposition 4.6. *If U is positive then $\nu a.U = U\frown^a$.*

Proof. By Proposition 3.6 and Lemma 4.4.

Lemma 4.7. *If U is positive then so is $\nu a.U$.*

Proof. Suppose U is positive. For every $u\backsim_A \propto U$, by Theorem 3.14 $A \subseteq \mathit{supp}(U)$. The result follows by construction and by Corollary 3.20.

Lemma 4.8. *Suppose $u\backsim_A$ and $u'\backsim_{A'}$ are positive. Suppose $u\backsim_A \subseteq u'\backsim_{A'}$.*
Then $u\backsim_{A\setminus\{a\}} \subseteq u'\backsim_{A'\setminus\{a\}}$.

Proof. Suppose $u\leftsquigarrow_A \subseteq u'\leftsquigarrow_{A'}$. So there exists some $\pi \in fix(A')$ such that $u = \pi{\cdot}u'$. Also, by Proposition 3.10 $A' \subseteq A$.

Consider some $\tau{\cdot}u \in u\leftsquigarrow_{A\setminus\{a\}}$ where $\tau \in fix(A \setminus \{a\})$. Then $\tau{\cdot}u = (\tau \circ \pi){\cdot}u'$, and $\tau \circ \pi \in fix(A' \setminus \{a\})$.

Remark 4.9. We illustrate why positivity is necessary in Lemma 4.8. Note that $\{a\} = a\leftsquigarrow_{\{a\}} \subseteq b\leftsquigarrow_{\{c\}} = \mathbb{A}\setminus\{c\}$ but $\mathbb{A} = a\leftsquigarrow_{\varnothing} \not\subseteq b\leftsquigarrow_{\{c\}}$. However, $b\leftsquigarrow_{\{c\}}$ is not positive because $\{c\} \not\subseteq supp(b) = \{b\}$.

Lemma 4.10. *Suppose U is positive. Then:*

1. *If $u\leftsquigarrow_{A'} \propto \nu a.U$ then $a \notin A'$ and either $u\leftsquigarrow_{A'} \propto U$ or $u\leftsquigarrow_{A'\cup\{a\}} \propto U$.*
2. *If $u\leftsquigarrow_A \propto U$ then $u\leftsquigarrow_{A\setminus\{a\}} \subseteq \nu a.U$.*

Proof. For part 1, suppose $u\leftsquigarrow_{A'} \propto \nu a.U$. By Lemma 4.7 $\nu a.U$ is positive. By Corollary 3.18 $u\leftsquigarrow_{A'} = u\leftsquigarrow_{A\setminus\{a\}}$ for some $u\leftsquigarrow_A \propto U$.

For part 2, if $u\leftsquigarrow_A \propto U$ then direct from Definition 4.5 $u\leftsquigarrow_{A\setminus\{a\}} \subseteq \nu a.U$.

Proposition 4.11. *For* positive U, $supp(\nu a.U) = supp(U) \setminus \{a\}$.

Proof. From Theorem 3.14 and part 1 of Lemma 4.10.

Remark 4.12. Proposition 4.11 is not what it seems. To the reader familiar with nominal techniques it may look just like e.g. [15, Corollary 5.2] which is the corresponding result for $[a]x$.

But consider $U = (\mathbb{A} \times \mathbb{A}) \setminus \{(a, b)\}$. This has three maximal planes: $(c, c)\leftsquigarrow_{\varnothing}$, $(c, b)\leftsquigarrow_a$, and $(a, c)\leftsquigarrow_b$, and is not positive. Also $supp(U) = \{a, b\}$, yet $supp(\nu a.U) = \varnothing \neq \{b\}$. The reason for this 'discrepancy' is the observation made in the Introduction that $\nu a.U$ and U have the same type, whereas $[a]U$ and U do not. Proposition 4.11 is different, and uses planes.

Corollary 4.13. *Suppose U is positive. Then $\nu a.U = U$ if and only if $a\#U$.*

Proof. Suppose $\nu a.U = U$. By Proposition 4.11 $a\#U$. Conversely if $a\#U$ then by Theorem 3.14 $a \notin A$ for every $x\leftsquigarrow_A \propto U$ and $\nu a.U = U$ follows by construction.

Lemma 4.14. $X\circlearrowright^A\circlearrowright^B = X\circlearrowright^{A\cup B}$.

Proof. By routine properties of permutations, and Theorem 2.6.

Corollary 4.15. *Suppose U is positive. Then $\nu a.\nu b.U = \nu b.\nu a.U$.*

Proof. From Proposition 4.6 and Lemma 4.14.

Proposition 4.16. $\pi{\cdot}(X\circlearrowright^A) = (\pi{\cdot}X)\circlearrowright^{\pi\cdot A}$.

As a corollary, if $b\#X$ then $(b\ a){\cdot}(X\circlearrowright^a) = ((b\ a){\cdot}X)\circlearrowright^b$ and $\nu a.X = \nu b.(b\ a){\cdot}X$.

Proof. The first part is by calculations on permutations, or by equivariance [13, Corollary 4.6]. The corollary follows using Proposition 4.11 and Theorem 2.6.

5 Nominal languages

Definition 5.1. Write $\mathbb{A}^*$ for the set of finite (possibly empty) strings of atoms.

k, l, m will range over elements of $\mathbb{A}^*$. We write kl for the concatenation of k and l. We write $[]$ for the empty string.

$\mathbb{A}^*$ is a nominal set with permutation action $\pi \cdot k = \pi(k_1) \dots \pi(k_n)$ where k_i is the ith element of k. Also, $supp(k) = \{k_1, \dots, k_n\}$ (the atoms in k).

A **nominal language** is a positive subset of $\mathbb{A}^*$. $\mathcal{K}$, $\mathcal{L}$ will range over languages.

An innocuous extension of Definition 5.1 is to use $(\mathbb{A} \cup \Sigma)^*$ for some Σ.

Definition 5.2. The union of two languages $\mathcal{K} \cup \mathcal{L}$ is their sets union.

Proposition 5.3. *Suppose X_i is positive for every $i \in I$. Suppose $\bigcup supp(X_i)$ is finite. Then $\bigcup X_i$ is positive. As a corollary, if $\mathcal{K}$ and $\mathcal{L}$ are languages then so is $\mathcal{K} \cup \mathcal{L}$.*

Proof. From Theorem 3.14 and Corollary 3.20.

Theorem 5.4. $(\nu a.\mathcal{K}) \cup (\nu a.\mathcal{L}) = \nu a.(\mathcal{K} \cup \mathcal{L})$.

Proof. Suppose $m\acute{}_C \propto (\nu a.\mathcal{K}) \cup (\nu a.\mathcal{L})$. Using Corollary 3.18 either $m\acute{}_C \propto \nu a.\mathcal{K}$ or $m\acute{}_C \propto \nu a.\mathcal{L}$; suppose without loss of generality $m\acute{}_C \propto \nu a.\mathcal{K}$. By part 1 of Lemma 4.10 $m\acute{}_C = k\acute{}_{A \setminus \{a\}}$ for $k\acute{}_A \propto \mathcal{K}$. Then $m\acute{}_C \subseteq \nu a.(\mathcal{K} \cup \mathcal{L})$ by part 2 of Lemma 4.10. The reverse inclusion follows by similar reasoning.

Definition 5.5. Write $u\acute{}_A \propto^S Z$ when $u\acute{}_A \propto Z$ and $supp(u) \cap S \subseteq A$.
We may omit brackets, writing e.g. $u\acute{}_A \propto^a Z$ for $u\acute{}_A \propto^{\{a\}} Z$.

Remark 5.6. $ac\acute{}_a \propto^b \{ax \mid x \in \mathbb{A} \setminus \{a\}\}$ and $ab\acute{}_a \not\propto^b \{ax \mid x \in \mathbb{A} \setminus \{a\}\}$. An analogy with λ-terms: $\lambda c.ac$ avoids name-clash with b whereas $\lambda b.ab$ does not. Think of $u\acute{}_A$ in Definition 5.5 as $\nu b_1 \dots \nu b_n.\{u\}$ where $\{b_1, \dots, b_n\} = supp(u) \setminus A$. 'Avoiding clash with S' means choosing a representative u such that $b_i \notin S$ for $1 \leq i \leq n$. In Definition 5.7, the superscript $supp(\mathcal{L})$ and $supp(\mathcal{K}) \cup supp(k)$ are *generalised capture-avoidance* conditions. More on this in Remark 5.8.

Definition 5.7. Define **composition** of languages $\mathcal{K} \circ \mathcal{L}$ by:

$$\mathcal{K} \circ \mathcal{L} = \bigcup \{kl\acute{}_{A \cup B} \mid k\acute{}_A \propto^{supp(\mathcal{L})} \mathcal{K},\ l\acute{}_B \propto^{supp(\mathcal{K}) \cup supp(k)} \mathcal{L}\}$$

Recall that here, kl denotes list concatenation. In words: $\mathcal{K} \circ \mathcal{L}$ is a capture-avoiding composition of the maximal planes of $\mathcal{K}$ and $\mathcal{L}$. For example, if $\mathcal{K} = \{a\}$ and $\mathcal{L} = \mathbb{A}$ then we can take $k = a$, $A = \{a\}$, $l = b$ and $B = \varnothing$ to calculate that $\mathcal{K} \circ \mathcal{L} = \{ax \mid x \in \mathbb{A} \setminus \{a\}\} = ab\acute{}_a$.

Remark 5.8. The 'definition' $\{kl\leftsquigarrow_{A\cup B} \mid k\leftsquigarrow_A \propto \mathcal{K},\ l\leftsquigarrow_B \propto \mathcal{L}\}$ would put the string aba in $(\nu b.ab)\circ \nu b.b$. This is undesired behaviour because $\nu b.b$ should avoid clash with the name a free in $\nu b.ab$. Similarly the 'ordinary' notion of composition $\{kl \mid k\in\mathcal{K},\ l\in\mathcal{L}\}$ does not avoid capture and delivers incorrect results.

Remark 5.9. The mention of $supp(k)$ in $l\leftsquigarrow_B \propto^{supp(\mathcal{K})\cup supp(k)} \mathcal{L}$ tells l to avoid name-clash with atoms used in k; if we wrote $l\leftsquigarrow_B \propto^{supp(\mathcal{K})} \mathcal{L}$ then composition would *deallocate* fresh names in k before executing l. Thus, Definition 5.7 does not put abb in $(\nu b.ab)\circ(\nu b.b)$, because of the mention of $supp(k)$.

Lemma 5.10. *$\mathcal{K}\circ\mathcal{L}$ is positive.*

Proof. By assumption $\mathcal{K}$ and $\mathcal{L}$ are positive. Suppose $k\leftsquigarrow_A \propto \mathcal{K}$ and $l\leftsquigarrow_B \propto \mathcal{L}$. By assumption $A\subseteq supp(k)$ and $B\subseteq supp(l)$ and it is a fact that therefore $A\cup B\subseteq supp(k)\cup supp(l) = supp(kl)$. Furthermore by Theorem 3.14 $A\subseteq supp(\mathcal{K})$ and $B\subseteq supp(\mathcal{L})$. The result follows by Corollary 3.20.

Theorem 5.11. *1. If $a\#\mathcal{L}$ then $(\nu a.\mathcal{K})\circ\mathcal{L} = \nu a.(\mathcal{K}\circ\mathcal{L})$.*
2. If $b\#\mathcal{K}$ then $\mathcal{K}\circ\nu b.\mathcal{L} = \nu b.(\mathcal{K}\circ\mathcal{L})$.

Proof (Sketch proof). We consider only the first part. Suppose $m\leftsquigarrow_C \propto \nu a.(\mathcal{K}\circ\mathcal{L})$. By definition and using part 1 of Lemma 4.10 and Corollary 3.18 there exist $k\leftsquigarrow_A \propto^{supp(\mathcal{L})} \mathcal{K}$ and $l\leftsquigarrow_B \propto^{supp(\mathcal{K})\cup supp(k)} \mathcal{L}$ such that $m\leftsquigarrow_C = kl\leftsquigarrow_{(A\cup B)\setminus\{a\}}$.

By part 1 of Lemma 4.10 $k\leftsquigarrow_{A\setminus\{a\}} \propto^{supp(\mathcal{L})\setminus\{a\}} \nu a.\mathcal{K}$. By Theorem 3.14 $a\notin B$ so $(A\cup B)\setminus\{a\} = (A\setminus\{a\})\cup B$. It follows by definition that $kl\leftsquigarrow_{(A\cup B)\setminus\{a\}} \subseteq (\nu a.\mathcal{K})\circ\mathcal{L}$.

The reverse inclusion follows by similar reasoning.

Lemma 5.12. *Suppose $\mathcal{K}$ and $\mathcal{L}$ are languages. Suppose $supp(\mathcal{K})\cup supp(\mathcal{L})\subseteq C$. Then $\mathcal{K}\circ\mathcal{L} = \{kl\leftsquigarrow_{A\cup B} \mid k\leftsquigarrow_A \propto^C \mathcal{K},\ l\leftsquigarrow_B \propto^{C\cup supp(k)} \mathcal{L}\}$.*

Proof. By Proposition 2.8 and Lemma 3.2.

Lemma 5.13. $(\mathcal{K}\circ\mathcal{L})\circ\mathcal{M} = \mathcal{K}\circ(\mathcal{L}\circ\mathcal{M})$.

Proof (Sketch proof). Set $C = supp(\mathcal{K})\cup supp(\mathcal{L})\cup supp(\mathcal{M})$. It is a fact that $supp(\mathcal{K}\circ\mathcal{L})\subseteq C$ and $supp(\mathcal{L}\circ\mathcal{K})\subseteq C$.[3]

By Corollary 3.18 and Lemma 5.12 if $n\leftsquigarrow_D \propto (\mathcal{K}\circ\mathcal{L})\circ\mathcal{M}$ then $n\leftsquigarrow_D = n'm\leftsquigarrow_{D'\cup C}$ for some $n'\leftsquigarrow_{D'} \propto^C \mathcal{K}\circ\mathcal{L}$ and some $m\leftsquigarrow_C \propto^{C\cup supp(n')} M$. Similarly, $n'\leftsquigarrow_{D'} = kl\leftsquigarrow_{A\cup B}$ for some $k\leftsquigarrow_A \propto^C \mathcal{K}$ and $l\leftsquigarrow_B \propto^{C\cup supp(k)} \mathcal{L}$.

By renaming k and l appropriately we may assume that $n' = kl$. It is a fact that $supp(n') = supp(k)\cup supp(l)$. It follows that

$$(\mathcal{K}\circ\mathcal{L})\circ M = \bigcup\{klm\leftsquigarrow_{A\cup B\cup C} \mid k\leftsquigarrow_A \propto^C \mathcal{K},\ l\leftsquigarrow_B \propto^{C\cup supp(k)} \mathcal{L},\ m\leftsquigarrow_C \propto^{C\cup supp(k)\cup supp(l)} \mathcal{M}\}.$$

The result follows.

[3] We can deduce this by direct calculations or by construction and using Theorem 3.14.

Theorem 5.14. *If* $\bigcup supp(\mathcal{L}_i)$ *is finite then* $\mathcal{K} \circ \bigcup \mathcal{L}_i = \bigcup(\mathcal{K} \circ \mathcal{L}_i)$ *and* $(\bigcup \mathcal{L}_i) \circ \mathcal{K} = \bigcup(\mathcal{L}_i \circ \mathcal{K})$.

Proof. We consider only the first part; the proof of the second part is similar. Set $G = supp(\mathcal{K}) \cup \bigcup supp(\mathcal{L}_i)$. We prove two subset inclusions.

- *Proof of the left-to-right subset inclusion.* Choose some $k\langle\!\!\cdot\!\!\rangle_A \propto^G \mathcal{K}$. Suppose $n\langle\!\!\cdot\!\!\rangle_D \propto^{G \cup supp(k)} \bigcup \mathcal{L}_i$. From Theorem 3.16 $n\langle\!\!\cdot\!\!\rangle_D \propto^{G \cup supp(k)} \mathcal{L}_i$ for some i. By Lemma 5.12 $kn\langle\!\!\cdot\!\!\rangle_{A \cup D} \subseteq \bigcup(\mathcal{K} \circ \mathcal{L}_i)$.
- *Proof of the right-to-left subset inclusion.* Suppose $n\langle\!\!\cdot\!\!\rangle_D \propto^G \bigcup(\mathcal{K} \circ \mathcal{L}_i)$. By Lemma 5.12 we may take $n = kl$ and $D = A \cup B$ for some $k\langle\!\!\cdot\!\!\rangle_A \propto^G \mathcal{K}$ and some i and $l\langle\!\!\cdot\!\!\rangle_B \propto^{G \cup supp(k)} \mathcal{L}_i$. Using Corollary 3.11 we can deduce that $l\langle\!\!\cdot\!\!\rangle_B \subseteq l\langle\!\!\cdot\!\!\rangle_{B'} \propto^{G \cup supp(k)} \bigcup \mathcal{L}_i$ for some $B' \subseteq B$. Since $A \cup B' \subseteq A \cup B$ we can conclude that $n\langle\!\!\cdot\!\!\rangle_D \subseteq kl\langle\!\!\cdot\!\!\rangle_{A \cup B'} \subseteq \mathcal{K} \circ \bigcup \mathcal{L}_i$.

Definition 5.15. Define $\mathcal{O} = \varnothing$ (the empty set). Define $\mathcal{I} = \{[]\}$ (recall from Definition 5.1 that [] is the empty string).

Definition 5.16. Define $\mathcal{K}^0 = \mathcal{I}$ and $\mathcal{K}^{i+1} = \mathcal{K}^i \circ \mathcal{K}$. Define $\mathcal{K}^* = \bigcup_i \mathcal{K}^i$.

Lemma 5.17. $\mathcal{O} \circ \mathcal{K} = \mathcal{O} = \mathcal{K} \circ \mathcal{O}$ *and* $\mathcal{I} \circ \mathcal{K} = \mathcal{K} = \mathcal{K} \circ \mathcal{I}$.

Theorem 5.18. $\mathcal{O}$ *and* $\mathcal{I}$ *are languages. Also, the set of languages is closed under* $\mathcal{K} \cup \mathcal{L}$, $va.\mathcal{L}$, $\mathcal{K} \circ \mathcal{L}$, *and* $\mathcal{K}^*$.

Proof. That $\mathcal{O}$ and $\mathcal{I}$ are languages is easy to verify. The case of $\mathcal{K} \cup \mathcal{L}$ is from Proposition 5.3; that of $va.\mathcal{L}$ is from Lemma 4.7; that of $\mathcal{K} \circ \mathcal{L}$ is from Lemma 5.10.

$supp(\mathcal{K}^i) \subseteq supp(\mathcal{K})$ can be verified by calculations or by equivariance [13, Theorem 4.7]. The case of $\mathcal{K}^*$ follows using Lemma 5.10 and Proposition 5.3.

6 Nominal Kleene algebra

We can use a nominal algebra style axiomatisation [14] to synthesise what we have proved so far as an extension of Kleene algebras (Definition 6.2 and Theorem 6.4). 'Nominal Kleene algebra' should be read tongue-in-cheek; we have no completeness proof like [20]. This is future work.

Definition 6.1. Call $x \in |\mathsf{X}|$ **equivariant** when $supp(x) = \varnothing$, thus $\forall \pi. \pi \cdot x = x$.

Call a function $f \in |\mathsf{X}| \to |\mathsf{Y}|$ **equivariant** when $\pi \cdot f(x) = f(\pi \cdot x)$ for all $x \in |\mathsf{X}|$ and all permutations π.

Definition 6.2. A **nominal Kleene algebra** is a tuple $\mathbb{X} = (|\mathbb{X}|, +, \cdot, ^*, 0, 1, \mathcal{V})$ of:

- A **nominal carrier set** $|\mathbb{X}|$.
- Equivariant functions $+$ and $\cdot$ from $||\mathbb{X}|| \times ||\mathbb{X}||$ to $||\mathbb{X}||$. We usually omit $\cdot$, writing e.g. XY for $X \cdot Y$.

$$
\begin{array}{rlrl}
X+(Y+Z) &= (X+Y)+Z & X+Y &= Y+X \\
X+0 &= X & X+X &= X \\
X(YZ) &= (XY)Z & & \\
1X &= X & X1 &= X \\
X(Y+Z) &= XY+XZ & (X+Y)Z &= XZ+YZ \\
0X &= 0 & X0 &= 0 \\
1+X(X^*) &\leq X^* & 1+X^*X &\leq X^* \\
XY \leq Y &\Rightarrow X^*Y \leq Y & YX \leq Y &\Rightarrow Y(X^*) \leq Y
\end{array}
$$

$$
\begin{array}{rlrl}
a\#X \Rightarrow & \nu a.X = X & & \nu a.\nu b.X = \nu b.\nu a.X \\
\nu a.X + \nu a.Y &= \nu a.(X+Y) & a\#Y \Rightarrow & (\nu a.X)Y = \nu a.(XY) \\
b\#X \Rightarrow & \nu a.X = \nu b.(b\ a)\cdot X & b\#X \Rightarrow & X(\nu b.Y) = \nu b.(XY)
\end{array}
$$

Fig. 1: Axioms of nominal Kleene algebra

- An equivariant function * from $||\mathbb{X}||$ to $||\mathbb{X}||$.
- Equivariant elements $0 \in ||\mathbb{X}||$ and $1 \in ||\mathbb{X}||$.
- An equivariant function $\mathcal{V}$ from $\mathbb{A} \times ||\mathbb{X}||$ to $||\mathbb{X}||$.

such that for all $X, Y, Z \in ||\mathbb{X}||$ and all $a, b \in \mathbb{A}$ the conditions in Figure 1 hold.

The upper axioms are the standard axioms of a Kleene algebra [20]. Here (as standard) we write $r \leq s$ as shorthand for $r + s = s$. Note that these axioms are not purely equational (so Kleene algebra is not, depending on terminology, actually algebraic), and the class of Kleene algebras forms a *quasi-variety*. This will not matter to us in this paper.

The axioms on the lower lines describe behaviour of name-restriction.

Remark 6.3. Note that $\mathcal{V}a.0 = 0$ and $\mathcal{V}a.1 = 1$ follow from the axiom $a\#X \Rightarrow \mathcal{V}a.X = X$, because it is a fact that $a\#0$ and $a\#1$.

Theorem 6.4. *Languages form a nominal Kleene algebra if we interpret $+$ as $\cup$, $\cdot$ as $\circ$, $\mathcal{V}$ as ν, and 0 and 1 as $\mathcal{O}$ and $\mathcal{I}$ respectively.*

Proof. We consider each axiom in turn:

- It is a fact that $\mathcal{K} \cup (\mathcal{L} \cup \mathcal{M}) = (\mathcal{K} \cup \mathcal{L}) \cup \mathcal{M}$, and $\mathcal{K} \cup \mathcal{L} = \mathcal{L} \cup \mathcal{K}$. It is also a fact that $\mathcal{K} \cup \mathcal{O} = \mathcal{K}$ and $\mathcal{K} \cup \mathcal{K} = \mathcal{K}$.
- $\mathcal{K} \circ (\mathcal{L} \circ \mathcal{M}) = (\mathcal{K} \circ \mathcal{L}) \circ \mathcal{M}$ by Lemma 5.13.
- $\mathcal{K} \circ (\mathcal{L} \cup \mathcal{M}) = (\mathcal{K} \circ \mathcal{L}) \cup (\mathcal{K} \circ \mathcal{M})$ and $(\mathcal{L} \cup \mathcal{M}) \circ \mathcal{K} = (\mathcal{L} \circ \mathcal{K}) \cup (\mathcal{L} \circ \mathcal{K})$ are by Theorem 5.14.
- $\mathcal{O} \circ \mathcal{K} = \mathcal{O} = \mathcal{K} \circ \mathcal{O}$ and $\mathcal{I} \circ \mathcal{K} = \mathcal{K} = \mathcal{K} \circ \mathcal{I}$ by Lemma 5.17.
- The four axioms for $\mathcal{K}^*$ follow using Theorem 5.14. To use some jargon, our denotation is **$*$-continuous** [19].
- If $a\#\mathcal{K}$ then $\nu a.\mathcal{K} = \mathcal{K}$ is by Corollary 4.13.
- $(\nu a.\mathcal{K}) \cup (\nu a.\mathcal{K}) = \nu a.(\mathcal{K} \cup \mathcal{L})$ is by Theorem 5.4.
- $(\nu a.\mathcal{K}) \circ (\nu a.\mathcal{L}) = \nu a.(\mathcal{K} \circ \mathcal{L})$ is by Theorem 5.11.
- $\nu a.\nu b.\mathcal{K} = \nu b.\nu a.\mathcal{K}$ is by Corollary 4.15.
- $b\#\mathcal{K} \Rightarrow \nu a.\mathcal{K} = \nu b.(b\ a)\cdot\mathcal{K}$ is by Proposition 4.16.

7 $\mathcal{L} \circ \mathbb{A}$ is equal to $\mathcal{L} \otimes \mathbb{A}$

Nominal sets have an *atoms tensor product* $\mathsf{X} \otimes \mathbb{A}$ ([27] or [13, Definition 9.27]) given by $\mathsf{X} \otimes \mathbb{A} = \{(x, a) \mid x \in \mathsf{X},\ a \in \mathbb{A},\ a\#x\}$. This has an obvious generalisation: $\mathsf{X} \otimes \mathsf{Y} = \{(x, y) \mid x \in \mathsf{X},\ y \in \mathsf{Y},\ supp(x) \cap supp(y) = \varnothing\}$. We can view $\circ$ as another (less obvious) generalisation of $\otimes$, as follows:

Proposition 7.1. *If $supp(\mathcal{K}) = \varnothing$ then $\mathcal{K} \circ \mathbb{A} = \mathcal{K} \otimes \mathbb{A}$ and $\mathbb{A} \circ \mathcal{K} = \mathbb{A} \otimes \mathcal{K}$, where we treat $\mathcal{K}$ as a nominal set with underlying set itself.*

Thus, composition of languages generalises $\otimes$. $\text{-} \otimes \mathbb{A}$ is left-adjoint to atoms-abstraction $[\mathbb{A}]\text{-}$ [13, Theorem 9.30]. By Proposition 7.1, so is $\text{-} \circ \mathbb{A}$. It remains to investigate the futher properties of $\text{-} \circ \mathbb{A}$.

8 Conclusions

Name-generation has long been a motivation for nominal techniques.

Odersky in [24] and Pitts and Stark [25] studied name-generation, and this was in the background thinking of the first author's and Pitts's development of Fraenkel-Mostowski/nominal sets. FreshML included a name-generating construct [29] which was a precursor of Fernández and the first author augmenting nominal terms and nominal rewriting explicitly with name generation $И a.t$ [5]; Pitts added a similar construct $\nu a.t$ to system T [26]. The axioms for αa in Figure 1 are of the same family.

A very abstract semantic study of name-generation is the *abstractive functions* considered in [11]. This influenced [12], where much machinery used in this paper was introduced. Abramski *et al.* give a concrete games semantics to the nu-calculus in nominal sets [1]: ideas here and in [12] also appear there, including Definition 3.1 (see e.g. Definition 2.7 of [30]).

There exist denotations for dynamic allocation using atoms-abstraction $[\mathbb{A}]\text{-}$, typically written δ in presheaf presentations. Examples are coalgebraic semantics for the π-calculus using δ (see e.g. [6, Subsection 2.2] or [2, Subsection 5.2]), the name-generation monad of FreshML [28, $F_{\langle\langle \mathtt{name} \rangle\rangle \tau}$, page 38]. We can also include the $X \restriction Y$ construct of nominal games from [1], which is in the same spirit and used in similar ways. In these examples name-generation exists at its own distinct level; in programming terms this corresponds to carrying around an explicit context of known 'fresh' names.

νa (Definition 4.5) is different because it places binding on a level with union $\cup$ and composition $\circ$: a language $\mathcal{L}$ is just a set of traces, not under a monad and not a set of α-equivalence classes of sets of traces. Thus we must work harder because freshness must be 'decrypted', but this buys us an appealingly simple model. A language really *is* just a set—as in the classical case of regular languages, without names and binding. That explicit context of known 'fresh' names is not explicitly necessary in the mathematical models we build.

One can raise the question of decidability of equality and inclusion between (subclasses of) languages, and automata. To consider such questions we need to match the developments of this paper with an automata-theoretic counterpart.

One well-studied notion of finite automaton with names and allocation is *history-dependent (HD-)automata* [23]. The correspondence to coalgebras over presheaves/nominal sets is considered in [3]. Investigation of the languages of HD-automata and the link with *finite-memory automata* [18] has shown that HD-automata are still essentially finite-memory machines [4]. However, the finite-support property of nominal sets corresponds to an idea of 'finitely but unboundedly many'. In FreshML, a type system in [10] first tried to restrict generation of fresh names and later in [28] the programming language appeared without such restrictions but the denotation used a monad to keep track of generated fresh names. Similarly, we would expect acceptors for languages from Section 5 to either impose bounds on support (if they are to be finite), or to be in the style of e.g. pushdown automata.

Most recently, *fresh register automata* have also been proposed, explicitly as an automaton model of names and fresh name generation [31]. It remains to investigate these in connection with this work.

We note in Remark 5.9 a 'deallocating' variant of composition $\mathcal{K} \circ \mathcal{L}$. There is a rich design space here to be studied in future work.

Nominal sets have further structure. We can model when a process omits a name (e.g. 'junk' in the π-calculus; a channel name that is not emitted yet occurs in the syntax of the term) using a freshness constraint: $X_{\#a} = \{x \in X \mid a\#X\}$.

Note that ν is not the $И$-quantifier introduced by the first author with Pitts in [15]. For instance, $X \subseteq \nu a.X$ is a fact, whereas $\phi(x) \Rightarrow Иx.\phi(x)$ is in general false. It *is* possible to define a version of $И$ acting on languages, given by $\mathsf{n}a.X = \{x \in X \mid Иb.(b\ a)\cdot x \in X\}$. We do not believe that n and ν are interdefinable and investigating them is future work.

Our models do not include negation; this is also future work.

Acknowledgements. Thanks to Emilio Tuosto and three anonymous referees. Supported by VICI grant 639.073.501 of the NWO and grant RYC-2006-002131 at the Polytechnic University of Madrid.

References

1. Samson Abramsky, Dan R. Ghica, Andrzej S. Murawski, C.-H. Luke Ong, and Ian D. B. Stark, *Nominal games and full abstraction for the nu-calculus*, Proceedings of the 19th IEEE Symposium on Logic in Computer Science (LICS 2004), IEEE Computer Society Press, 2004, pp. 150–159.
2. Marcello Bonsangue and Alexander Kurz, *Pi-calculus in logical form*, Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007), IEEE Computer Society Press, 2007, pp. 303–312.
3. Vincenzo Ciancia and Ugo Montanari, *Symmetries, local names and dynamic (de)-allocation of names*, Information and Computation (2010), In press.
4. Vincenzo Ciancia and Emilio Tuosto, *A novel class of automata for languages on infinite alphabets*, Tech. Report CS-09-003, University of Leicester, UK, 2009.
5. Maribel Fernández and Murdoch J. Gabbay, *Nominal rewriting with name generation: abstraction vs. locality*, Proceedings of the 7th ACM SIGPLAN International Symposium on Principles and Practice of Declarative Programming (PPDP 2005), ACM Press, July 2005, pp. 47–58.
6. Marcelo Fiore, Eugenio Moggi, and Davide Sangiorgi, *A fully-abstract model for the π-calculus (extended abstract)*, Proceedings of the 11th IEEE Symposium on Logic in Computer Science (LICS 1996), IEEE Computer Society Press, 1996, pp. 43–54.

7. Marcelo Fiore and Sam Staton, *Comparing operational models of name-passing process calculi*, Information and Computation **204** (2006), no. 4, 524–560.
8. Marcelo Fiore and Daniele Turi, *Semantics of name and value passing*, Proceedings of the 16th IEEE Symposium on Logic in Computer Science (LICS 2001), IEEE Computer Society Press, 2001, pp. 93–104.
9. Nissim Francez, C. A. R. Hoare, Daniel J. Lehmann, and Willem P. de Roever, *Semantics of nondeterminism, concurrency, and communication*, Journal of Computer and System Sciences **19** (1979), no. 3, 290–308.
10. Murdoch J. Gabbay, *A Theory of Inductive Definitions with alpha-Equivalence*, Ph.D. thesis, University of Cambridge, UK, March 2001.
11. _____, *A General Mathematics of Names*, Information and Computation **205** (2007), no. 7, 982–1011.
12. _____, *A study of substitution, using nominal techniques and Fraenkel-Mostowski sets*, Theoretical Computer Science **410** (2009), no. 12-13, 1159–1189.
13. _____, *Foundations of nominal techniques: logic and semantics of variables in abstract syntax*, Bulletin of Symbolic Logic (2010), In press.
14. Murdoch J. Gabbay and Aad Mathijssen, *Nominal universal algebra: equational logic with names and binding*, Journal of Logic and Computation **19** (2009), no. 6, 1455–1508.
15. Murdoch J. Gabbay and Andrew M. Pitts, *A New Approach to Abstract Syntax with Variable Binding*, Formal Aspects of Computing **13** (2001), no. 3–5, 341–363.
16. Fabio Gadducci, Marino Miculan, and Ugo Montanari, *About permutation algebras, (pre)sheaves and named sets*, Higher-Order and Symbolic Computation **19** (2006), no. 2-3, 283–304.
17. Gerard J. Holzmann, *The spin model checker: Primer and reference manual*, Addison-Wesley Professional, September 2003.
18. Michael Kaminski and Nissim Francez, *Finite-memory automata*, Theoretical Computer Science **134** (1994), no. 2, 329363.
19. Dexter Kozen, *On induction vs. $*$-continuity*, Proceedings of the Logic of Programs Workshop, Lecture notes in computer science, vol. 131, Springer, 1982, pp. 167–176.
20. Dexter Kozen, *A completeness theorem for Kleene algebras and the algebra of regular events*, Information and Computation **110** (1994), no. 2, 366390.
21. Marino Miculan and Ivan Scagnetto, *A framework for typed HOAS and semantics*, Principles and Practice of Declarative Programming, 5th International ACM SIGPLAN Symposium (PPDP 2003), ACM, 2003, pp. 184–194.
22. Eugenio Moggi, *Notions of computation and monads*, Information and Computation **93** (1991), no. 1, 55–92.
23. Ugo Montanari and Marco Pistore, *Pi-calculus, structured coalgebras and minimal hd-automata*, MFCS 2000 (Mogens Nielsen and Branislav Roman, eds.), Lecture Notes in Computer Science, no. 1983, Springer, 2000.
24. Martin Odersky, *A functional theory of local names*, Proceedings of the 21st Annual ACM Symposium on Principles of Programming Languages (POPL'94), ACM Press, 1994, p. 4859.
25. A. M. Pitts and I. D. B. Stark, *Observable properties of higher order functions that dynamically create local names, or: What's new?*, Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science (MFCS 1993), Lecture Notes in Computer Science, vol. 711, Springer, 1993, pp. 122–141.
26. Andrew M. Pitts, *Nominal system T*, Proceedings of the 37th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL 2010), ACM Press, January 2010, pp. 159–170.
27. Andrew M. Pitts and Murdoch J. Gabbay, *A Metalanguage for Programming with Bound Names Modulo Renaming*, Proceedings of the 5th international conference on the mathematics of program construction (MPC 2000), Lecture Notes in Computer Science, vol. 1837, Springer, December 2000, pp. 230–255.
28. Mark R. Shinwell and Andrew M. Pitts, *On a monadic semantics for freshness*, Theoretical Computer Science **342** (2005), no. 1, 28–55.
29. Mark R. Shinwell, Andrew M. Pitts, and Murdoch J. Gabbay, *FreshML: Programming with Binders Made Simple*, Proceedings of the 8th ACM SIGPLAN International Conference on Functional Programming (ICFP 2003), vol. 38, ACM Press, August 2003, pp. 263–274.
30. Nikos Tzevelekos, *Nominal game semantics*, Ph.D. thesis, Oxford, 2008.
31. _____, *Fresh-register automata*, Proceedings of the 38th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL 2011), ACM Press, January 2011.