Completeness and Incompleteness in Nominal Kleene Algebra

Dexter Kozen* Konstantinos Mamouras* Alexandra Silva†

November 14, 2014

Abstract

Gabbay and Ciancia (2011) presented a nominal extension of Kleene algebra as a framework for trace semantics with dynamic allocation of resources, along with a semantics consisting of nominal languages. They also provided an axiomatization that captures the behavior of the scoping operator and its interaction with the Kleene algebra operators and proved soundness over nominal languages. In this paper we show that the axioms are complete and describe the free language models.

1 Introduction

Nominal sets are a convenient framework for handling name generation and binding. They were introduced by Gabbay and Pitts [3] as a mathematical model of name binding and α-conversion.

Nominal extensions of classical automata theory have been recently explored [1], motivated by the increasing need for tools for languages over infinite alphabets. These play a role in various areas, including XML document processing, cryptography, and verification. An XML document can be seen as a tree with labels from the (infinite) set of all unicode strings that can appear as attribute values. In cryptography, infinite alphabets are used as *nonces*, names used only once in cryptographic communications to prevent replay attacks. In software verification, infinite alphabets are used for references, objects, pointers, and function parameters.

In this paper, we focus on axiomatizations of regular languages and how these can be lifted in the presence of a binding operator and an infinite alphabet of names. This work builds on the recent work of Gabbay and Ciancia [5],

*Computer Science, Cornell University, Ithaca, New York 14853-7501, USA. `http://www.cs.cornell.edu/~kozen/`, `http://www.cs.cornell.edu/~mamouras/`. This work was done while visiting Radboud University Nijmegen.

†Intelligent Systems, Radboud University Nijmegen, Postbus 9010, 6500 GL Nijmegen, The Netherlands. `http://alexandrasilva.org`.

who presented a nominal extension of Kleene algebra as a framework for trace semantics with dynamic allocation of resources, along with a semantics consisting of *nominal languages*. Gabbay and Ciancia also provided an axiomatization that captures the behavior of the scoping operator and its interaction with the usual Kleene algebra operators. They proved soundness of their axiomatization over nominal languages, but left open the question of completeness. In this paper we tackle this problem.

Intuitively, the challenge behind showing completeness is twofold. On the one hand, one needs to find the appropriate (language) model, or in other words, the free model. On the other hand, there is an inherent need to find an appropriate *normal form* for a given expression. Normal forms are a vehicle to completeness: two expressions are equivalent if they can be reduced to the same normal form, and the axioms are complete if they enable us to derive normal forms for all expressions.

Our approach is modular. We show that under the right definition of a language model, one can prove completeness by first transforming the expression to another expression for which only the usual Kleene algebra axioms are needed. The steps of the transformation make use of the usual axioms of Kleene algebra along with axioms proposed by Gabbay and Ciancia for the scoping operator.

We also show that the axioms are not complete for the standard language model proposed by Gabbay and Ciancia. We explain exactly what the problem is with their original language model, which contains what they called *non-maximal planes*. This technical difference will be clear later in the paper. We also show that the axioms are not complete for summation models in which the scoping operator is interpreted as a summation operator over a finite set.

Interestingly, in devising the proof of completeness, we have developed a technique that might be useful in other completeness proofs. More precisely, we have made use of the well-known fact that the Boolean algebra generated by finitely many regular sets consists of regular sets and is atomic. Hence, expressions can be written as sums of atoms. This is crucial in obtaining the normal form. To our knowledge this has not been used before in completeness proofs.

The paper is organized as follows. In §2 we recall basic material on Kleene algebra (KA), nominal sets, and the nominal extension of KA (NKA) by Gabbay and Ciancia. In §3 we discuss the possible language models, starting with the original one proposed in [5] and then introducing two new ones: our own alternative language model and three summation models. We give a precise description of the difference between the two language models. In §4 we present our main result on completeness. The proof is given in four steps: *exposing bound variables*, *scope configuration*, *canonical choice of bound variables*, and *semilattice identities*. The last step uses a novel technique based on the atomicity of the the Boolean algebra generated by finitely many regular sets. In §5 we present concluding remarks and directions for future work.

2 Background

2.1 Kleene Algebra (KA)

Kleene algebra is the algebra of regular expressions. Regular expressions are normally interpreted as regular sets of strings, but there are other useful interpretations: binary relation models used in programming language semantics, the $(\min, +)$ algebra used in shortest path algorithms, models consisting of convex sets used in computational geometry, and many others.

A *Kleene algebra* is any structure $(K, +, \cdot, ^*, 0, 1)$ where K is a set, + and $\cdot$ are binary operations on K, * is a unary operation on K, and 0 and 1 are constants, satisfying the following axioms:

$$\begin{array}{ll}
x + (y + z) = (x + y) + z & x(yz) = (xy)z \\
x + y = y + x & 1x = x1 = x \\
x + 0 = x + x = x & x0 = 0x = 0 \\
x(y + z) = xy + xz & (x + y)z = xz + yz \\
1 + xx^* \le x^* & y + xz \le z \;\Rightarrow\; x^*y \le z \\
1 + x^*x \le x^* & y + zx \le z \;\Rightarrow\; yx^* \le z
\end{array}$$

where we define $x \le y$ iff $x + y = y$. The axioms above not involving * are succinctly stated by saying that the structure is an idempotent semiring under $+, \cdot, 0$, and 1, the term *idempotent* referring to the axiom $x + x = x$. Due to this axiom, the ordering relation $\le$ is a partial order. The axioms for * together say that x^*y is the $\le$-least z such that $y + xz \le z$ and yx^* is the $\le$-least z such that $y + zx \le z$.

2.2 Group Action

A *group action* of a group G on a set X is a map $G \times X \to X$ (written as juxtaposition) such that $\pi(\rho x) = (\pi\rho)x$. For $A \subseteq X$, define the subgroups

$$\begin{aligned}
\mathsf{fix}\, A &= \{\pi \in G \mid \forall x \in A\ \pi x = x\} \\
\mathsf{Fix}\, A &= \{\pi \in G \mid \forall x \in A\ \pi x \in A\} = \{\pi \in G \mid \pi A = A\} \\
\mathsf{fix}\, x &= \mathsf{fix}\{x\} = \mathsf{Fix}\{x\}.
\end{aligned}$$

Thus $\mathsf{fix}\, A$ is the subgroup of all elements of G that fix A pointwise and $\mathsf{Fix}\, A$ is the subgroup of all elements of G that fix A setwise.

2.3 Nominal Sets

Let $\mathbb{A}$ be a countably infinite set of *atoms* and let G be the group of all finite permutations of G (permutations generated by transpositions $(a\ b)$). The group G acts on $\mathbb{A}$ in the obvious way. If X is another set on which G acts, we say that $A \subseteq \mathbb{A}$ *supports* $x \in X$ if

$$\mathsf{fix}\,A \subseteq \mathsf{fix}\,x.$$

An element $x \in X$ has *finite support* if there is a finite set $A \subseteq \mathbb{A}$ that supports x. A *nominal set* is a set X with a group action of G on X such that every element of X has finite support.

Lemma 2.1 *For $A, B \subseteq \mathbb{A}$ such that $A \cup B \neq \mathbb{A}$,*

$$\mathsf{fix}(A \cap B) = \mathsf{fix}\,A \vee \mathsf{fix}\,B,$$

the least subgroup of G containing both $\mathsf{fix}\,A$ and $\mathsf{fix}\,B$.

Proof. Clearly $\mathsf{fix}\,A \subseteq \mathsf{fix}(A \cap B)$ and $\mathsf{fix}\,B \subseteq \mathsf{fix}(A \cap B)$, therefore the right-to-left inclusion holds. For the left-to-right inclusion, write any element of $\mathsf{fix}(A \cap B)$ as a finite product of transpositions $(a\ b)$ with $a, b \notin A \cap B$. For each such transposition, either $a, b \notin A$, in which case $(a\ b) \in \mathsf{fix}\,A$, or $a, b \notin B$, in which case $(a\ b) \in \mathsf{fix}\,B$, or one of a, b is in $A - B$ and the other is in $B - A$, in which case we can write $(a\ b) = (a\ c)(b\ c)(a\ c)$ where c is any element in $\mathbb{A} - (A \cup B)$. We have written the element of $\mathsf{fix}(A \cap B)$ as a product of elements of $\mathsf{fix}\,A$ and $\mathsf{fix}\,B$. $\square$

Thus if A and B are finite and support x, then so does $A \cap B$. It follows that if x is finitely supported, there is a smallest set that supports it, which we call $\mathsf{supp}\,x$.

Lemma 2.2 *A supports x iff πA supports πx. In particular, $\mathsf{supp}\,\pi x = \pi\,\mathsf{supp}\,x$.*

Proof. For the first statement,

$$\begin{aligned} A \text{ supports } x &\Leftrightarrow \mathsf{fix}\,A \subseteq \mathsf{fix}\,x \\ &\Leftrightarrow \pi(\mathsf{fix}\,A)\pi^{-1} \subseteq \pi(\mathsf{fix}\,x)\pi^{-1} \\ &\Leftrightarrow \mathsf{fix}\,\pi A \subseteq \mathsf{fix}\,\pi x \\ &\Leftrightarrow \pi A \text{ supports } \pi x. \end{aligned}$$

For the second statement,

$$\begin{aligned} \mathsf{supp}\,\pi x &= \bigcap \{A \mid A \text{ supports } \pi x\} \\ &= \bigcap \{\pi A \mid \pi A \text{ supports } \pi x\} \\ &= \pi(\bigcap \{A \mid \pi A \text{ supports } \pi x\}) \\ &= \pi(\bigcap \{A \mid A \text{ supports } x\}) \\ &= \pi\,\mathsf{supp}\,x. \end{aligned}$$

$\square$

Lemma 2.3 *For $x \in X$, $\mathsf{fix}\,\mathsf{supp}\,x \subseteq \mathsf{fix}\,x \subseteq \mathsf{Fix}\,\mathsf{supp}\,x$. Both inclusions can be strict.*

Proof. The first inclusion is just the statement that $\mathsf{supp}\,x$ supports x. For the second,

$$\pi x = x \Rightarrow \pi\,\mathsf{supp}\,x = \mathsf{supp}\,\pi x = \mathsf{supp}\,x.$$

Both inclusions can be strict, as can be seen from the element $x = (a+b)c$ in a nominal semiring. The transposition $(a\ b)$ is in $\mathsf{fix}\,x - \mathsf{fix}\{a,b,c\}$ due to the commutativity of addition, and the transposition $(a\ c)$ is in $\mathsf{Fix}\{a,b,c\} - \mathsf{fix}\,x$. $\square$

One can show that $\mathsf{fix}\,\mathsf{supp}\,x$ is a normal subgroup of $\mathsf{Fix}\,\mathsf{supp}\,x$, therefore also of $\mathsf{fix}\,x$, and the quotient groups $\mathsf{fix}\,x/\,\mathsf{fix}\,\mathsf{supp}\,x$ and $\mathsf{Fix}\,\mathsf{supp}\,x/\,\mathsf{fix}\,\mathsf{supp}\,x$ are isomorphic to permutation groups on $\mathsf{supp}\,x$. The latter is isomorphic to S_n, the symmetric group on n letters, where n is the cardinality of $\mathsf{supp}\,x$. The former is a subgroup of the latter and characterizes those permutations of $\mathsf{supp}\,x$ that preserve x.

2.4 Syntax of Nominal KA

Expressions over an alphabet Σ of primitive letters are

$$e ::= a \in \Sigma \mid e+e \mid ee \mid e^* \mid 0 \mid 1 \mid \nu a.e.$$

The scope of the binding νa in $\nu a.e$ is e. Notational convention: the binding operator νa is of lower precedence than product but higher precedence than sum; thus in products, scopes extend as far to the right as possible. Example:

$$\nu a.ab\ \nu b.ba = \nu a.(ab\ \nu b.(ba)) \qquad \text{and not} \qquad (\nu a.ab)(\nu b.ba)$$

The set of expressions over Σ is denoted Exp_Σ.

A *ν-string* is an expression with no occurrence of +, *, or 0, and no occurrence of 1 except to denote the null string, in which case we use ε instead.

$$x ::= a \in \Sigma \mid xx \mid \varepsilon \mid \nu a.x$$

The set of ν-strings over Σ is denoted Σ^ν.

The *free variables* of expressions and ν-strings are defined inductively:

$$\mathsf{FV}(a) = \{a\} \qquad \mathsf{FV}(0) = \mathsf{FV}(1) = \varnothing \qquad \mathsf{FV}(e^*) = \mathsf{FV}(e)$$
$$\mathsf{FV}(e_1+e_2) = \mathsf{FV}(e_1e_2) = \mathsf{FV}(e_1) \cup \mathsf{FV}(e_2) \qquad \mathsf{FV}(\nu a.e) = \mathsf{FV}(e) - \{a\}.$$

The nominal axioms proposed by Gabbay and Ciancia [5] are:

$$\begin{aligned} &\nu a.(d+e) = \nu a.d + \nu a.e && a\#e \Rightarrow \nu b.e = \nu a.(a\ b)e \\ &\nu a.\nu b.e = \nu b.\nu a.e && a\#e \Rightarrow (\nu a.d)e = \nu a.de \\ &a\#e \Rightarrow \nu a.e = e && a\#e \Rightarrow e(\nu a.d) = \nu a.ed. \end{aligned} \qquad (1)$$

3 Models

3.1 Nominal KA

A *nominal KA* (NKA) over atoms $\mathbb{A}$ is a structure

$$(K, +, \cdot, {}^*, 0, 1, \nu)$$

such that K is a nominal set over atoms $\mathbb{A}$ and for all $\pi \in G$,

$$\begin{aligned} \pi(x+y) &= \pi x + \pi y & \qquad \pi(0) &= 0 \\ \pi(xy) &= (\pi x)(\pi y) & \pi(1) &= 1 \\ \pi(x^*) &= (\pi x)^* & \pi(\nu a.e) &= \nu(\pi a).\pi e, \end{aligned}$$

that is, the action of every $\pi \in G$ is an automorphism of K, and all the KA and nominal axioms are satisfied.

3.2 Nominal Language Model

Now we describe a nominal language interpretation $NL : \mathsf{Exp}_{\mathbb{A}} \to \mathcal{P}(\mathbb{A}^*)$ for each expression e that interprets expressions over $\mathbb{A}$ as certain subsets of $\mathbb{A}^*$. This is the language model of [5]. The definition is slightly nonstandard, as care must be taken when defining product to avoid capture.

First we give an intermediate interpretation $I : \mathsf{Exp}_{\mathbb{A}} \to \mathcal{P}(\mathbb{A}^\nu)$ of expressions as sets of ν-strings over $\mathbb{A}$. The regular operators $+$, $\cdot$, *, 0, and 1 have their usual set-theoretic interpretations, and

$$I(\nu a.e) = \{\nu a.x \mid x \in I(e)\} \qquad\qquad I(a) = \{a\}.$$

We maintain the scoping of ν-subexpressions in the ν-strings. Examples:

$$\begin{aligned} I(\nu a.a) &= \{\nu a.a\} \\ I(\nu a.\nu b.(a+b)) &= \{\nu a.\nu b.a, \nu a.\nu b.b\} \\ I(\nu a.(\nu b.ab)(a+b)) &= \{\nu a.(\nu b.ab)a, \nu a.(\nu b.ab)b\} \\ I(\nu a.(ab)^*) &= \{\nu a.\varepsilon, \nu a.ab, \nu a.abab, \nu a.ababab, \ldots\} \\ I((\nu a.ab)^*) &= \{\varepsilon, \nu a.ab, (\nu a.ab)(\nu a.ab), (\nu a.ab)(\nu a.ab)(\nu a.ab), \ldots\}. \end{aligned}$$

Now we describe the map $NL : \mathbb{A}^\nu \to \mathcal{P}(\mathbb{A}^*)$ on ν-strings. Given a ν-string x, first α-convert so that all bindings in x are distinct and different from all free variables in x, then delete all binding operators νa to obtain a string $x' \in \mathbb{A}^*$. For example,

$$(\nu a.ab)(\nu a.ab)(\nu a.ab)' = abcbdb.$$

Here we have α-converted to obtain $(\nu a.ab)(\nu c.cb)(\nu d.db)$, then deleted the binding operators to obtain $abcbdb$. The choice of variables in the α-conversion does not matter as long as they are distinct and different from the free variables.

Now we define for each ν-string x and expression e

$$NL(x) = \{\pi x' \mid \pi \in \mathsf{fix}\,\mathsf{FV}(x)\} \qquad\qquad NL(e) = \bigcup_{x \in I(e)} NL(x).$$

The set $NL(x)$ is the plane $x' \circlearrowleft_{\mathsf{FV}(x)}$ in the notation of [5]. Thus we let the bound variables range simultaneously over all possible values in $\mathbb{A}$ they could take on, as long as they remain distinct and different from the free variables, and we accumulate all strings obtained in this way. For example,

$NL((\nu a.ab)(\nu a.ab)(\nu a.ab)) = \{abcbdb \mid a, c, d \in \mathbb{A}$ distinct and different from $b\}$.

As mentioned, the choice of fresh variables in the α-conversion does not matter; thus

$$NL(x) = \{\pi y \mid \pi \in \mathsf{fix}\,\mathsf{FV}(x)\} \tag{2}$$

for any $y \in NL(x)$.

For $x, y \in \mathbb{A}^\nu$, write $x \equiv y$ if x and y are equivalent modulo the nominal axioms (1). The following lemma says that the nominal axioms alone are sound and complete for equivalence between ν-strings in the nominal language model.

Lemma 3.1 *For $x, y \in \mathbb{A}^\nu$, $x \equiv y$ if and only if $NL(x) = NL(y)$.*

Proof. Soundness (the left-to-right implication) holds because each nominal axiom preserves NL, as is not difficult to check. For completeness (the right-to-left implication), suppose $NL(x) = NL(y)$. We must have $\mathsf{FV}(x) = \mathsf{FV}(y)$, because if $a \in \mathsf{FV}(x) - \mathsf{FV}(y)$, then $NL(y)$ would contain a string with no occurrence of a, whereas all strings in $NL(x)$ contain an occurrence of a. Now α-convert x and y so that all bound variables are distinct and different from the free variables, and move the bound variables to the front, so that $x = \nu A.x'$ and $y = \nu B.y'$ for some $x', y' \in \mathbb{A}^*$. By (2), $y' = \pi x'$ for some $\pi \in \mathsf{fix}\,\mathsf{FV}(x) = \mathsf{fix}\,\mathsf{FV}(y)$, so $x = \pi y$, and $\pi y \equiv y$ by α-conversion. $\square$

Lemma 3.2 *For any $x \in \mathbb{A}^*$ and $A, B \subseteq \mathsf{FV}(x)$,*

$$A \subseteq B \Leftrightarrow NL(\nu A.x) \subseteq NL(\nu B.x).$$

In the notation of [5],

$$A \subseteq B \Leftrightarrow x \circlearrowleft_{B'} \subseteq x \circlearrowleft_{A'},$$

where $A' = \mathsf{FV}(x) - A$ and $B' = \mathsf{FV}(x) - B$.

Proof. If $A \subseteq B$, then $\mathsf{fix}\,A' \subseteq \mathsf{fix}\,B'$, therefore

$$NL(\nu A.x) = \{\pi x \mid \pi \in \mathsf{fix}\,A'\} \subseteq \{\pi x \mid \pi \in \mathsf{fix}\,B'\} = NL(\nu B.x).$$

Conversely, if $a \in A - B$, then $x[b/a] \in NL(\nu A.x) - NL(\nu B.x)$, where b is any element of $\mathbb{A} - \mathsf{FV}(x)$. $\square$

Lemma 3.3 *Let $y \in NL(e)$ and $A \subseteq \mathsf{FV}(y)$ maximal such that $NL(\nu A.y) \subseteq NL(e)$ (in the notation of [5], this is $y\circlearrowleft_{A'} \propto NL(e)$, where $A' = \mathsf{FV}(y) - A$). Then $\nu A.y \in I(e)$, and $\nu A.y$ is the unique ν-string up to nominal equivalence for which this is true.*

Remark This is the essential content of [5, Theorem 3.16]. This is important for us because it says that the set $NL(e)$ uniquely determines the maximal elements of $I(e)$ up to nominal equivalence (Lemma 3.4 below).

Proof. Let $x_1, \ldots, x_n \in I(e)$ be all ν-strings such that $y \in NL(x_i)$. There are only finitely many of these. Then

$$NL(\nu A.y) \subseteq NL(x_1) \cup \cdots \cup NL(x_n) \subseteq NL(e).$$

Using the nominal axioms (1), we can move the quantification in each x_i to the front of the string and α-convert so that the quantifier-free part is y. This is possible because $y \in NL(x_i)$. Thus we can assume without loss of generality that each $x_i = \nu A_i.y$ for some $A_i \subseteq \mathsf{FV}(y)$.

Let $z \in NL(\nu A.y)$ such that $(\mathsf{FV}(z) - \mathsf{FV}(\nu A.y)) \cap \mathsf{FV}(\nu A_i.y) = \varnothing$, $1 \le i \le n$. Since

$$NL(\nu A.y) \subseteq NL(x_1) \cup \cdots \cup NL(x_n) = NL(\nu A_1.y) \cup \cdots \cup NL(\nu A_n.y),$$

we must have $z \in NL(\nu A_i.y)$ for some i. But then $\mathsf{FV}(\nu A.y), \mathsf{FV}(\nu A_i.y) \subseteq \mathsf{FV}(z)$ and $\mathsf{FV}(\nu A_i.y) \subseteq \mathsf{FV}(\nu A.y)$ by choice of z, therefore $A \subseteq A_i$. Since A was maximal, $A = A_i$. □

Let $\widehat{I}(e) = \{x \in I(e) \mid NL(x) \text{ is maximal in } NL(e)\}$.

Lemma 3.4 *$NL(e_1) = NL(e_2)$ if and only if $\widehat{I}(e_1) = \widehat{I}(e_2)$ modulo the nominal axioms* (1).

Proof. Suppose $NL(e_1) = NL(e_2)$. By Lemma 3.3, each $y \in NL(e_1)$ is contained in a unique maximal $NL(\nu A.y)$, and $\nu A.y \in \widehat{I}(e_1)$. As $NL(e_1) = NL(e_2)$, these planes are also contained in $NL(e_2)$. Similarly, the maximal planes of $NL(e_2)$ are contained in $NL(e_1)$. Since the two sets contain the same set of maximal planes, they must be equal, therefore $\widehat{I}(e_1) = \widehat{I}(e_2)$ modulo the nominal axioms.

For the reverse implication, note that

$$NL(e) = \bigcup_{x \in I(e)} NL(x) = \bigcup_{x \in \widehat{I}(e)} NL(x)$$

by the fact that every plane of e is contained in a maximal one. Then

$$NL(e_1) = \bigcup_{x \in \widehat{I}(e_1)} NL(x) = \bigcup_{x \in \widehat{I}(e_2)} NL(x) = NL(e_2).$$

□

3.3 Alternative Nominal Language Model

Let Σ and $\mathbb{A}$ be countably infinite disjoint sets. Letters $a, b, c, \ldots$ range over $\mathbb{A}$, $x, y, z, \ldots$ over Σ, and $u, v, w, \ldots$ over $(\Sigma \cup \mathbb{A})^*$. Quantification is only over Σ.

A *language* is a subset $A \subseteq (\Sigma \cup \mathbb{A})^*$ such that $\pi A = A$ for all $\pi \in G$. The set of languages is denoted $\mathcal{L}$.

The operations of nominal KA are defined on $\mathcal{L}$ as follows:

$$\begin{aligned}
A + B &= A \cup B \\
AB &= \{uv \mid u \in A,\ v \in B,\ \mathsf{FV}(u) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing\} \\
A^* &= \bigcup_n A^n \\
0 &= \varnothing \\
1 &= \{\varepsilon\} \\
\nu x.A &= \{w[a/x] \mid w \in A,\ a \in \mathbb{A} - \mathsf{FV}(w)\},\ x \in \Sigma.
\end{aligned}$$

Lemma 3.5 *The set $\mathcal{L}$ is closed under the operations of nominal KA.*

Proof. For sum, $\pi(\bigcup_n A_n) = \bigcup_n \pi A_n = \bigcup_n A_n$. For product,

$$\begin{aligned}
\pi(AB) &= \{\pi(uv) \mid u \in A,\ v \in B,\ \mathsf{FV}(u) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing\} \\
&= \{(\pi u)(\pi v) \mid u \in A,\ v \in B,\ \mathsf{FV}(\pi u) \cap \mathsf{FV}(\pi v) \cap \pi\mathbb{A} = \varnothing\} \\
&= \{uv \mid u \in \pi A,\ v \in \pi B,\ \mathsf{FV}(u) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing\} \\
&= (\pi A)(\pi B) = AB.
\end{aligned}$$

The case of A^* follows from the previous two cases. The cases of 0 and 1 are trivial. Finally, for $\nu x.A$, we have

$$\begin{aligned}
\pi(\nu x.A) &= \{\pi(w[a/x]) \mid w \in A,\ a \in \mathbb{A} - \mathsf{FV}(w)\} \\
&= \{(\pi w)[\pi a/x] \mid w \in A,\ a \in \mathbb{A} - \mathsf{FV}(w)\} \\
&= \{w[a/x] \mid \pi^{-1}w \in A,\ \pi^{-1}a \in \mathbb{A} - \mathsf{FV}(\pi^{-1}w)\} \\
&= \{w[a/x] \mid w \in \pi A,\ a \in \pi\mathbb{A} - \pi\mathsf{FV}(\pi^{-1}w)\} \\
&= \{w[a/x] \mid w \in A,\ a \in \mathbb{A} - \mathsf{FV}(w)\} \\
&= \nu x.A.
\end{aligned}$$

□

We can interpret nominal KA expressions as languages. The interpretation map $AL : \mathsf{Exp}_\Sigma \to \mathcal{L}$ is the unique homomorphism with respect to the above language operations such that $AL(x) = \{x\}$. Note that in this context, atoms $a \in \mathbb{A}$ do not appear in expressions or ν-strings.

Theorem 3.6 *The nominal axioms* (1) *hold in this model.*

Proof.

$$\begin{aligned}
AL(\nu x.(d+e)) &= \{w[a/x] \mid w \in AL(d+e),\ a \in \mathbb{A} - \mathsf{FV}(w)\} \\
&= \{w[a/x] \mid w \in AL(d) \cup AL(e),\ a \in \mathbb{A} - \mathsf{FV}(w)\} \\
&= \{w[a/x] \mid w \in AL(d),\ a \in \mathbb{A} - \mathsf{FV}(w)\} \\
&\qquad \cup \{w[a/x] \mid w \in AL(e),\ a \in \mathbb{A} - \mathsf{FV}(w)\} \\
&= AL(\nu x.d) \cup AL(\nu x.e)
\end{aligned}$$

$$\begin{aligned}
AL(\nu x.\nu y.e) &= \{w[a/x] \mid w \in AL(\nu y.e),\ a \in \mathbb{A} - \mathsf{FV}(w)\} \\
&= \{w[a/x] \mid w \in \{u[b/y] \mid u \in AL(e),\ b \in \mathbb{A} - \mathsf{FV}(u)\}, \\
&\qquad\qquad\qquad\qquad\qquad\qquad a \in \mathbb{A} - \mathsf{FV}(w)\} \\
&= \{u[b/y][a/x] \mid u \in AL(e),\ b \in \mathbb{A} - \mathsf{FV}(u),\ a \in \mathbb{A} - \mathsf{FV}(u[b/y])\} \\
&= \{u[a/x][b/y] \mid u \in AL(e),\ a \in \mathbb{A} - \mathsf{FV}(u),\ b \in \mathbb{A} - \mathsf{FV}(u[a/x])\} \\
&= AL(\nu y.\nu x.e).
\end{aligned}$$

For the remaining axioms, assume $x \notin \mathsf{FV}(e)$. Then

$$\begin{aligned}
AL(\nu x.e) &= \nu x.AL(e) \\
&= \{w[a/x] \mid w \in AL(e),\ a \in \mathbb{A} - \mathsf{FV}(w)\} \\
&= \{w \mid w \in AL(e),\ a \in \mathbb{A} - \mathsf{FV}(w)\} \qquad (3) \\
&= AL(e).
\end{aligned}$$

The equation (3) holds since $\mathsf{FV}(w) \cap \Sigma \subseteq \mathsf{FV}(e)$, therefore $x \notin \mathsf{FV}(w)$, so $w[a/x] = w$.

$$\begin{aligned}
AL(\nu y.e) &= \nu y.AL(e) \\
&= \{w[a/y] \mid w \in AL(e),\ a \in \mathbb{A} - \mathsf{FV}(w)\} \\
&= \{((x\ y)w)[a/x] \mid (x\ y)w \in AL((x\ y)e),\ a \in \mathbb{A} - \mathsf{FV}((x\ y)w)\} \\
&= \{w[a/x] \mid w \in AL((x\ y)e),\ a \in \mathbb{A} - \mathsf{FV}(w)\} \\
&= AL(\nu x.(x\ y)e).
\end{aligned}$$

$$\begin{aligned}
&AL((\nu x.d)e)\\
&= \{uv \mid u \in AL(\nu x.d),\ v \in AL(e),\ \mathsf{FV}(u) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing\}\\
&= \{uv \mid u \in \nu x.AL(d),\ v \in AL(e),\ \mathsf{FV}(u) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing\}\\
&= \{uv \mid u \in \{w[a/x] \mid w \in AL(d),\ a \in \mathbb{A} - \mathsf{FV}(w)\},\ v \in AL(e),\\
&\qquad\qquad \mathsf{FV}(u) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing\}\\
&= \{w[a/x]v \mid w \in AL(d),\ a \in \mathbb{A} - \mathsf{FV}(w),\ v \in AL(e),\\
&\qquad\qquad \mathsf{FV}(w[a/x]) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing\}\\
&= \{w[a/x]v \mid w \in AL(d),\ a \in \mathbb{A} - \mathsf{FV}(wv),\ v \in AL(e),\\
&\qquad\qquad \mathsf{FV}(w) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing\} \qquad (4)\\
&= \{u[a/x]v \mid u \in AL(d),\ v \in AL(e),\ \mathsf{FV}(u) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing,\\
&\qquad\qquad a \in \mathbb{A} - \mathsf{FV}(uv)\}\\
&= \{(uv)[a/x] \mid u \in AL(d),\ v \in AL(e),\ \mathsf{FV}(u) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing,\\
&\qquad\qquad a \in \mathbb{A} - \mathsf{FV}(uv)\}\\
&= \{w[a/x] \mid w \in \{uv \mid u \in AL(d),\ v \in AL(e),\ \mathsf{FV}(u) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing\},\\
&\qquad\qquad a \in \mathbb{A} - \mathsf{FV}(w)\}\\
&= \{w[a/x] \mid w \in AL(de),\ a \in \mathbb{A} - \mathsf{FV}(w)\}\\
&= \nu x.AL(de)\\
&= AL(\nu x.de).
\end{aligned}$$

All steps are straightforward except (4), which requires an argument. We condsider two cases: either $x \in \mathsf{FV}(w)$ or $x \notin \mathsf{FV}(w)$. In the former case, we argue that

$$\begin{aligned}
&a \in \mathbb{A} - \mathsf{FV}(w) \text{ and } \mathsf{FV}(w[a/x]) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing\\
&\Leftrightarrow a \in \mathbb{A} - \mathsf{FV}(wv) \text{ and } \mathsf{FV}(w) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing.
\end{aligned}$$

For the left-to-right implication, since $x \in \mathsf{FV}(w)$, we have $a \in \mathsf{FV}(w[a/x]) \cap \mathbb{A}$, and since $\mathsf{FV}(w[a/x]) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing$, it must be that $a \notin \mathsf{FV}(v)$, therefore $a \in \mathbb{A} - (\mathsf{FV}(w) \cup \mathsf{FV}(v)) = \mathbb{A} - \mathsf{FV}(wv)$. Also, $\mathsf{FV}(w) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing$, since $\mathsf{FV}(w[a/x]) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing$ and $\mathsf{FV}(w) \cap \mathbb{A} \subseteq \mathsf{FV}(w[a/x]) \cap \mathbb{A}$.

For the right-to-left implication, since $a \in \mathbb{A} - \mathsf{FV}(wv) = \mathbb{A} - (\mathsf{FV}(w) \cup \mathsf{FV}(v))$, we have $a \in \mathbb{A} - \mathsf{FV}(w)$ and $a \in \mathbb{A} - \mathsf{FV}(v)$, therefore

$$\begin{aligned}
&\mathsf{FV}(w[a/x]) \cap \mathsf{FV}(v) \cap \mathbb{A}\\
&= (\mathsf{FV}(w) \cup \{a\}) \cap \mathsf{FV}(v) \cap \mathbb{A}\\
&= (\mathsf{FV}(w) \cap \mathsf{FV}(v) \cap \mathbb{A}) \cup (\{a\} \cap \mathsf{FV}(v) \cap \mathbb{A})\\
&= \varnothing \cup \varnothing = \varnothing.
\end{aligned}$$

In the latter case ($x \notin \mathsf{FV}(w)$), the equation (4) reduces to

$$\begin{aligned}
&\{wv \mid w \in AL(d),\ a \in \mathbb{A} - \mathsf{FV}(w),\ v \in AL(e),\ \mathsf{FV}(w) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing\}\\
&= \{wv \mid w \in AL(d),\ a \in \mathbb{A} - \mathsf{FV}(wv),\ v \in AL(e),\ \mathsf{FV}(w) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing\},
\end{aligned}$$

which is clearly true, as a is irrelevant.

Similarly, $AL(e(\nu x.d)) = AL(\nu x.ed)$. $\square$

We can also define $I : \mathsf{Exp}_\Sigma \to \Sigma^\nu$ and $\widehat{I} : \mathsf{Exp}_\Sigma \to \Sigma^\nu$ exactly as in §3.2 for the nominal language model, with the modification that expressions are over Σ and not $\mathbb{A}$.

Lemma 3.7 $AL(e) = \bigcup_{w \in I(e)} AL(w)$.

Proof. This can be proved by a straightforward induction on the structure of e. We argue the case of products and binders explicitly.

$$\begin{aligned}
AL(e_1e_2) &= \{uv \mid u \in AL(e_1),\ v \in AL(e_2),\ \mathsf{FV}(u) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing\} \\
&= \{uv \mid u \in \bigcup_{p \in I(e_1)} AL(p),\ v \in \bigcup_{q \in I(e_2)} AL(q),\ \mathsf{FV}(u) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing\} \\
&= \bigcup_{\substack{p \in I(e_1) \\ q \in I(e_2)}} \{uv \mid u \in AL(p),\ v \in AL(q),\ \mathsf{FV}(u) \cap \mathsf{FV}(v) \cap \mathbb{A} = \varnothing\} \\
&= \bigcup_{\substack{p \in I(e_1) \\ q \in I(e_2)}} AL(pq) = \bigcup_{r \in I(e_1e_2)} AL(r).
\end{aligned}$$

$$\begin{aligned}
AL(\nu x.e) &= \nu x.AL(e) \\
&= \{w[a/x] \mid w \in AL(e),\ a \in \mathbb{A} - \mathsf{FV}(w)\} \\
&= \{w[a/x] \mid w \in \bigcup_{p \in I(e)} AL(p),\ a \in \mathbb{A} - \mathsf{FV}(w)\} \\
&= \bigcup_{p \in I(e)} \{w[a/x] \mid w \in AL(p),\ a \in \mathbb{A} - \mathsf{FV}(w)\} \\
&= \bigcup_{p \in I(e)} \nu x.AL(p) = \bigcup_{p \in I(e)} AL(\nu x.p) = \bigcup_{w \in I(\nu x.e)} AL(w).
\end{aligned}$$

$\square$

Lemma 3.8 *Every plane $AL(\nu A.w)$ in $AL(e)$ is maximal; that is, $I(e) = \widehat{I}(e)$.*

Proof. Replace each $x \in A$ in w with a distinct element of $\mathbb{A}$ to get w'. Then

$$AL(\nu A.w) = \{\pi w' \mid \pi \in G\}.$$

This is maximal, as all finite permutations of $\mathbb{A}$ are allowed. $\square$

Lemma 3.8 characterizes the key difference between the nominal language model of §3.2 and the alternative nominal language model of this section. It explains why the axioms are complete for the alternative model but not for the model of §3.2. In the model of §3.2, there are non-maximal planes, and these are "hidden" by the maximal planes, whereas this cannot happen in the alternative model, as all planes are maximal.

3.4 Summation Model over the Free KA

It is sound to interpret νx as a summation operator $\sum_{a \in F} e[a/x]$. Let K be the free KA on generators X (regular expressions over X modulo the KA axioms) and let $F \subseteq K$. For $x \in X$, interpret

$$\nu x.e = \sum_{a \in F} e[a/x] \qquad\qquad x\#e \Leftrightarrow x \notin \mathsf{FV}(e),$$

where $e[a/x]$ is the expression obtained by substituting a for x in e. This can be interpreted in any KA in which the sums exist; and in any KA when F is finite.

This is a sound interpretation, as all the nominal axioms are satisfied:

$$\begin{aligned}
&\sum_a (d+e)[a/x] = \sum_a d[a/x] + \sum_a e[a/x] \\
&\sum_a \sum_b e[b/y][a/x] = \sum_b \sum_a e[a/x][b/y] \\
&x \notin \mathsf{FV}(e) \Rightarrow \sum_a e[a/x] = e \\
&x \notin \mathsf{FV}(e) \Rightarrow \sum_a e[a/y] = \sum_a e[x/y][a/x] \\
&x \notin \mathsf{FV}(e) \Rightarrow (\sum_a d[a/x])e = \sum_a de[a/x] \\
&x \notin \mathsf{FV}(e) \Rightarrow e(\sum_a d[a/x]) = \sum_a ed[a/x].
\end{aligned}$$

3.5 Language Summation Model

In particular, let $\mathbb{A}$ be a set of letters, finite or infinite. We wish to interpret expressions as subsets of $\mathbb{A}^*$ by a map $L : \mathsf{Exp}_{\mathbb{A}} \to \mathcal{P}(\mathbb{A}^*)$. Let the regular operators have their usual set-theoretic interpretations, and let

$$L(\nu x.e) = \bigcup_{a \in \mathbb{A}} L(e[a/x]) \qquad\qquad L(a) = \{a\},\ a \in \mathbb{A}.$$

3.6 Summation Model over an Arbitrary KA K

More generally, let K be an arbitrary KA and let $K[X]$ be the set of polynomials over indeterminates X with coefficients in K. This is the direct sum (coproduct) of K with the free KA on generators X. Let $F \subseteq K$ be finite. Let $e \mapsto e[a/x]$ be the evaluation morphism that for $x \in X$ evaluates $e \in K[X]$ at $x = a$. We can interpret

$$\nu x.e = \sum_{a \in F} e[a/x] \qquad\qquad x\#e \Leftrightarrow e \in K[X - \{x\}] \Leftrightarrow e[0/x] = e.$$

This is also sound interpretation:

$$\begin{aligned}
&\sum_a (d+e)[a/x] = \sum_a d[a/x] + \sum_a e[a/x]\\
&\sum_a \sum_b e[b/y][a/x] = \sum_b \sum_a e[a/x][b/y]\\
&e[0/x] = e \Rightarrow \sum_a e[a/x] = e\\
&e[0/x] = e \Rightarrow \sum_a e[a/y] = \sum_a e[x/y][a/x]\\
&e[0/x] = e \Rightarrow (\sum_a d[a/x])e = \sum_a de[a/x]\\
&e[0/x] = e \Rightarrow e(\sum_a d[a/x]) = \sum_a ed[a/x].
\end{aligned}$$

In fact the set F can be a singleton $\{a\}$; in this case, $\nu x.e$ is just evaluation $e \mapsto e[a/x]$.

$$\begin{aligned}
&(d+e)[a/x] = d[a/x] + e[a/x]\\
&e[b/y][a/x] = e[a/x][b/y]\\
&e[0/x] = e \Rightarrow e[a/x] = e\\
&e[0/x] = e \Rightarrow e[a/y] = e[x/y][a/x]\\
&e[0/x] = e \Rightarrow (d[a/x])e = de[a/x]\\
&e[0/x] = e \Rightarrow e(d[a/x]) = ed[a/x].
\end{aligned}$$

4 Completeness

In this section we prove our main theorem:

Theorem 4.1 *The axioms of nominal Kleene algebra are sound and complete for the equational theory of nominal Kleene algebras and for the equational theory of the alternative language interpretation of* §3.3.

We thus show that if two nominal KA expressions e_1 and e_2 are equivalent in the alternative language interpretation of §3.3 in the sense that $AL(e_1) = AL(e_2)$, then e_1 and e_2 are provably equivalent in the axiomatization of Gabbay and Ciancia [5]. This says that the alternative language model of §3.3 is the free nominal KA. This is not true of Gabbay and Ciancia's language model presented in §3.2, as the inequality $a \leq \nu a.a$ holds in the language model of §3.2 but not in the summation models. Neither is it true of the summation models of §3.5 and §3.6, as $\nu a.aa \leq \nu a.\nu b.ab$ holds in the summation models but not in the language model. However, it is true of Gabbay and Ciancia's language model if one restricts to closed terms, as the closed terms of the language models of §3.2 and §3.3 are the same.

We show that every expression can be put into a particular canonical form that will allow us to apply the KA axioms to prove equivalence. This construction will consist of several steps: *exposing bound variables*, *scope configuration*, *canonical choice of bound variables*, and *determining semilattice identities*. Each step will involve a construction that is justified by the axioms.

For the purposes of exposition, we write $\underset{a}{(}\, e \,\underset{a}{)}$ instead of $\nu a.e$ so that it is easier to see the scope boundaries. In this notation, the nominal axioms take the following form:

$$
\begin{aligned}
&\nu a.(d+e) = \nu a.d + \nu a.e && \underset{a}{(}\, d+e \,\underset{a}{)} = \underset{a}{(}\, d \,\underset{a}{)} + \underset{a}{(}\, e \,\underset{a}{)} && (5)\\
&\nu a.\nu b.e = \nu b.\nu a.e && \underset{a}{(}\underset{b}{(}\, e \,\underset{b}{)}\underset{a}{)} = \underset{b}{(}\underset{a}{(}\, e \,\underset{a}{)}\underset{b}{)} && (6)\\
&a\#e \Rightarrow \nu a.e = e && a\#e \Rightarrow \underset{a}{(}\, e \,\underset{a}{)} = e && (7)\\
&a\#e \Rightarrow \nu b.e = \nu a.(a\ b)e && a\#e \Rightarrow \underset{b}{(}\, e \,\underset{b}{)} = \underset{a}{(}(a\ b)e \,\underset{a}{)} && (8)\\
&a\#e \Rightarrow (\nu a.d)e = \nu a.de && a\#e \Rightarrow \underset{a}{(}\, d \,\underset{a}{)}\, e = \underset{a}{(}\, de \,\underset{a}{)} && (9)\\
&a\#e \Rightarrow e(\nu a.d) = \nu a.ed && a\#e \Rightarrow e\, \underset{a}{(}\, d \,\underset{a}{)} = \underset{a}{(}\, ed \,\underset{a}{)}\,. && (10)
\end{aligned}
$$

We remark that writing scope boundaries of ν-expressions as letters $\underset{a}{(}$ and $\underset{a}{)}$ is merely a notational convenience. Although it appears to allow us to violate the invariant that starred expressions and ν-expressions are mutually well-nested, in reality it does not, as all our transformations are justified by the axioms, which maintain this invariant.

4.1 Exposing Bound Variables

A *ν^*-string* is a string of

(i) letters a,

(ii) well-nested scope delimiters $\underset{a}{(}$ and $\underset{a}{)}$, and

(iii) starred expressions e^* whose bodies e are (inductively) sums of ν^*-strings.

We say that the bound variables of a ν^*-string are *exposed* if

(i) the first and last occurrence of each bound variable occur at the top level in the scope of their binding operator,[1] and

(ii) the bound variables of all ν^*-strings in the bodies of starred subexpressions are (inductively) exposed.

[1] "Top level" means not inside a starred subexpression. Inside a starred expression e^*, "top level" means not inside a starred subexpression of e.

A typical ν^*-string is

$$\underset{a}{(}\,\underset{b}{(}\,abb(ab\,\underset{a}{(}\,ab\,\underset{a}{)}+b\,\underset{b}{(}\,ba\,\underset{b}{)})^*ba\,\underset{b}{)}\,\underset{a}{)}\,.$$

The bound variables are exposed in this expression because the first and last occurrences of a and b occur at the top level. Inside the starred subexpression, the bound variables in the two ν^*-strings are exposed because there are no starred subexpressions.

Lemma 4.2 *Every expression can be written as a sum of ν^*-strings whose bound variables are exposed.*

Proof. It is straightforward to see how to use the nominal axiom (5) in the left-to-right direction and the distributivity and 0 and 1 laws of Kleene algebra to write every expression as a sum of ν^*-strings.

Exposing the bound variables is a little more difficult. It may appear at first glance that one can simply unwind e^* as $1 + e + ee^*e$ and then unwind the starred subexpressions of e inductively, but this is not enough. For example,

$$\begin{aligned}(a+b)^* &= 1 + a + b + (a+b)(a+b)^*(a+b)\\ &= 1 + a + b + a(a+b)^*a + a(a+b)^*b + b(a+b)^*a + b(a+b)^*b,\end{aligned}$$

and the subexpression $a(a+b)^*a$ does not satisfy (i). The following more complicated expression is needed:

$$\begin{aligned}(a+b)^* &= 1 + a + b + aa^*a + bb^*b + ab + ba && (11)\\ &\quad + a^*ab + aa^*ba^*a + baa^*a + bb^*ba + bb^*ab^*b + abb^*b && (12)\\ &\quad + aa^*abb^*b + aa^*b(a+b)^*ab^*b + aa^*b(a+b)^*ba^*a && (13)\\ &\quad + bb^*a(a+b)^*ab^*b + bb^*a(a+b)^*ba^*a + bb^*baa^*a && (14)\end{aligned}$$

Line (11) covers strings containing no a's or no b's or one of each. Line (12) covers strings containing one a and two or more or more b's or one b and two or more or more a's. Lines (13) and (14) cover strings containing at least two a's and at least two b's.

For the general construction, we first argue the case of $(a_1 + \cdots + a_n)^*$. Write down all strings containing either zero, one, or two occurrences of each letter. For each such string, insert a starred subexpression in each gap between adjacent letters. The body of the starred expression inserted into a gap will be the sum of all letters a such that the gap falls between two occurrences of a.

For example, the second term of (13) is obtained from the string $abab$. There are three gaps, into which we insert the indicated starred expressions:

$$\begin{array}{ccccccc} a & & b & & a & & b\\ & \uparrow & & \uparrow & & \uparrow & \\ & a^* & & (a+b)^* & & b^* & \end{array}$$

In the first gap we inserted a^* because the gap falls between two occurrences of a but not between two occurrences of b. In the second gap we inserted $(a+b)^*$ because the gap falls between two occurrences of a and two occurrences of b.

This construction covers all strings whose first and last occurrences of each letter occur in the order specified by the original string before the insertion. If a letter occurs twice before the insertion, then after the insertion those two occurrences are the first and last, and they occur at the top level. If a letter occurs once before the insertion, then that is the only occurrence after the insertion, and it is at the top level. If a letter does not occur at all before the insertion, then it does not occur after.

For the general case e^*, we first perform the construction inductively on all starred subexpressions of e, writing $e^* = (e_1 + \cdots + e_n)^*$ where each top-level ν^*-string e_i satisfies (i) and (ii). Now take the sum constructed above for $(a_1 + \cdots + a_n)^*$ and substitute e_i for a_i in all terms. This gives an expression of the desired form. $\square$

4.2 Scope Configuration

For this part of the construction, we first α-convert using (8) to make all bound variables distinct and different from any free variable. This is called the *Barendregt variable convention*.

Now we transform each ν^*-string to ensure that every top-level left delimiter $\underset{a}{(}$ occurs immediately to the left of a free occurrence of the variable a that it binds:

$$\cdots \underset{a}{(}\, a \cdots \underset{b}{(}\, b \cdots \underset{c}{(}\, c \cdots \underset{c}{)} \cdots \underset{b}{)} \cdots \underset{a}{)} \cdots \tag{15}$$

That occurrence is at the top level due to the preprocessing step of §4.1. We do this without changing the order of any occurrences of variables in the string, but we may change the order of quantification.

Starting at the left end of the string, scan right, looking for top-level left delimiters. For all top-level left delimiters that we see, push them to the right as long as we do not encounter a variable bound by any of them. Stop when such a variable is encountered. For example,

$$\cdots \underset{a}{(} \cdots \underset{b}{(} \cdots \underset{c}{(} \cdots b \cdots \underset{c}{)} \cdots \underset{b}{)} \cdots \underset{a}{)} \cdots \;\Rightarrow\; \cdots \underset{a}{(}\underset{b}{(}\underset{c}{(}\, b \cdots \underset{c}{)} \cdots \underset{b}{)} \cdots \underset{a}{)} \cdots$$

Here we are using the nominal axiom (10) in the right-to-left direction to skip over letters and starred expressions. If such a variable is encountered, it will be at the top level because of the preprocessing step of §4.1.

In this example, we must keep the $\underset{b}{(}$ to the left of that occurrence of b, but we wish to move the $\underset{a}{(}$ and $\underset{c}{(}$ past the b. The c can be moved in using (10), but to move the a in, we must exchange the order of quantification of a and b. To do this, we push the corresponding right delimiter of b up to the right delimiter of

a using the nominal axiom (9) in the left-to-right direction.

$$\cdots \underset{a}{(}\,\underset{b}{(}\,\underset{c}{(}\, b \cdots \underset{c}{)} \cdots \underset{b}{)} \cdots \underset{a}{)} \cdots \quad \Rightarrow \quad \cdots \underset{a}{(}\,\underset{b}{(}\,\underset{c}{(}\, b \cdots \underset{c}{)} \cdots \underset{b}{)}\,\underset{a}{)} \cdots$$

This is always possible, as there is no free occurrence of b to the right of the $\underset{b}{)}$ due to the Barendregt variable convention. Now we can exchange the order of quantification using the nominal axiom (6).

$$\cdots \underset{a}{(}\,\underset{b}{(}\,\underset{c}{(}\, b \cdots \underset{c}{)} \cdots \underset{b}{)}\,\underset{a}{)} \cdots \quad \Rightarrow \quad \cdots \underset{b}{(}\,\underset{a}{(}\,\underset{c}{(}\, b \cdots \underset{c}{)} \cdots \underset{a}{)}\,\underset{b}{)} \cdots$$

This allows us to move the a and c in past the $\underset{b}{(}$ and continue.

$$\cdots \underset{b}{(}\,\underset{a}{(}\,\underset{c}{(}\, b \cdots \underset{c}{)} \cdots \underset{a}{)}\,\underset{b}{)} \cdots \quad \Rightarrow \quad \cdots \underset{b}{(}\, b\, \underset{a}{(}\,\underset{c}{(} \cdots \underset{c}{)} \cdots \underset{a}{)}\,\underset{b}{)} \cdots$$

When looking for the first occurrence of a free variable bound to a left delimiter, perhaps no free occurrence is encountered before seeing a right delimiter. In this case there is no free occurrence of the variable in the scope of the binding, so we can just forget the binding altogether.

$$\begin{aligned} \cdots \underset{a}{(} \cdots \underset{b}{(} \cdots \underset{c}{(} \cdots \underset{c}{)} \cdots b \cdots \underset{b}{)} \cdots \underset{a}{)} \cdots \quad &\Rightarrow \quad \cdots \underset{a}{(}\,\underset{b}{(}\,\underset{c}{(}\,\underset{c}{)} \cdots b \cdots \underset{b}{)} \cdots \underset{a}{)} \cdots \\ &\Rightarrow \quad \cdots \underset{a}{(}\,\underset{b}{(} \cdots b \cdots \underset{b}{)} \cdots \underset{a}{)} \cdots \end{aligned}$$

This uses the nominal axiom (7).

If there exists a free occurrence of a inside a scope $\underset{a}{(} \cdots \underset{a}{)}$, then the leftmost one occurs at the top level due to the construction of §4.1. Thus, when we are done, any remaining left delimiters $\underset{a}{(}$ in the string occur immediately to the left of a free occurrence of a that is bound to that delimiter, as illustrated in (15).

Now we finish up the construction by moving the right delimiters to the left as far as possible without exchanging order of quantification. Because of the preprocessing step of §4.1, the rightmost occurrence of any variable quantified at the top level occurs at the top level. Thus every right delimiter $\underset{a}{)}$ occurs either immediately to the right of an occurrence of a bound to that delimiter or immediately to the right of another right delimiter $\underset{b}{)}$ with smaller scope.

At this point we have transformed the expression so that every ν^*-string satisfies the following properties:

(i) every ν-subformula is of the form $\nu a.ae$; that is, the leftmost symbol of every scope is a variable bound by that scope; and

(ii) the rightmost boundary of every scope is as far to the left as possible, subject to (i).

The position of the scope delimiters is canonical, because scopes are as small as possible: the left delimiters are as far to the right as they can possibly be, and the right delimiters are as far to the left as they can possibly be given the positions of the left delimiters. It follows that if two expressions are equivalent, then they generate the same ν-strings up to renaming of bound variables.

4.3 Canonical Choice of Bound Variables

Now we would like to transform the expression so that the bound variables are chosen in a canonical way. This will ensure that if two expressions are equivalent, then they generate the same ν-strings, not just up to renaming of bound variables, but absolutely. This part of the construction will thus relax the Barendregt variable convention, so that variables can be bound more than once and can occur both bound and free in a string.

Choose a set of variables disjoint from the free variables of the expression and order them in some arbitrary but fixed order $a_0, a_1, \ldots$. Moving through the expression from left to right, maintain a stack of variable names corresponding to the scopes we are currently in. When a left scope delimiter $\underset{a}{(}$ is encountered, and we are inside the scope of n ν-formulas, the variables $a_0, \ldots, a_{n-1}$ will be on the stack. We rename the bound variable a to a_n using the nominal axiom (8) for α-conversion and push a_n onto the stack. When a right scope delimiter is encountered, we pop the stack. This construction guarantees that every ν-string generated by the expression satisfies:

- For every symbol in the string, if the symbol occurs in the scope of n nested ν-expressions, then those expressions bind variables $a_0, \ldots, a_{n-1}$ in that order from outermost to innermost scope.

It follows that two semantically equivalent expressions so transformed generate exactly the same set of ν-strings.

4.4 Determining Semilattice Identities

After transforming e_1 and e_2 by the above construction, we know that if e_1 and e_2 are equivalent, then they generate the same sets of ν-strings; that is, $I(e_1) = I(e_2)$. Now we wish to show that any two such expressions can be proved equivalent using the KA and nominal axioms in conjunction with the following congruence rule for ν-formulas:

$$\frac{e_1 = e_2}{\nu a.e_1 = \nu a.e_2}. \tag{16}$$

In order to do this, there is one more issue that must be resolved. Let us first assume for simplicity that e_1 and e_2 are of ν-depth one; that is, they only contain bindings of one variable a. There may be several subexpressions in e_1 and e_2 of the form $\nu a.d$, but all with the same variable a. We will relax this restriction later.

Any substring of the form $\nu a.x$ of a ν-string generated by e_1 or e_2 must be generated by a subexpression of the form $\nu a.d$. However, there may be several different subexpressions of this form, and the string $\nu a.x$ could be generated by more than one of them. In general, the sets of ν-strings generated by the ν-subexpressions could satisfy various semilattice identities, and we may have to know these identities in order to prove equivalence.

For example, consider the two expressions $c_1 + c_2$ and $d_1 + d_2 + d_3$, where

$$\begin{aligned} c_1 &= \nu a.a(aa)^* & d_1 &= \nu a.a(aaa)^* \\ c_2 &= \nu a.aa(aa)^* & d_2 &= \nu a.aa(aaa)^* \\ & & d_3 &= \nu a.aaa(aaa)^* \end{aligned} \tag{17}$$

(c_i generates strings with i mod 2 a's and d_i generates strings with i mod 3 a's). Both $c_1 + c_2$ and $d_1 + d_2 + d_3$ generate all nonempty strings of a's, but in different ways. If $c_1 + c_2$ occurs in e_1 and $d_1 + d_2 + d_3$ occurs in e_2, we would have to know that they are equivalent to prove the equivalence of e_1 and e_2.

To determine all semilattice identities such as $c_1 + c_2 = d_1 + d_2 + d_3$ that hold among the ν-subexpressions, we express every ν-subexpression in e_1 or e_2 as a sum of atoms of the Boolean algebra on sets of ν-strings generated by these ν-subexpressions. In the example above, the atoms of the generated Boolean algebra are

$$b_i = \nu a.a^i(a^6)^*,\ 1 \le i \le 6 \tag{18}$$

(b_i generates strings with i mod 6 a's). Rewriting the expressions (17) as sums of atoms, we would obtain

$$\begin{aligned} c_1 &= b_1 + b_3 + b_5 & d_1 &= b_1 + b_4 \\ c_2 &= b_2 + b_4 + b_6 & d_2 &= b_2 + b_5 \\ & & d_3 &= b_3 + b_6. \end{aligned}$$

The equivalences are provable in pure KA plus the nominal axiom (5). Then $c_1 + c_2$ and $d_1 + d_2 + d_3$ become

$$\begin{aligned} c_1 + c_2 &= (b_1 + b_3 + b_5) + (b_2 + b_4 + b_6) \\ d_1 + d_2 + d_3 &= (b_1 + b_4) + (b_2 + b_5) + (b_3 + b_6), \end{aligned}$$

which are clearly equivalent.

Now we observe that any ν-string $\nu a.x$ generated by e_1 or e_2 is generated by exactly one atom. Moreover, if $\nu a.f$ is an atom and $\nu a.x \in I(\nu a.f)$, and if $\nu a.x$ is generated by $\nu a.f$ in the context $u(\nu a.x)v \in I(\nu a.e_1)$, then for any other $\nu a.y \in I(\nu a.f)$, we have $u(\nu a.y)v \in I(\nu a.e_1)$ as well. This says that we may treat $\nu a.f$ as atomic. In fact, once we have determined the atoms, if we like we may replace each atom $\nu a.f$ by a single letter $a_{\nu a.f}$ in e_1 and e_2, and the resulting expressions are equivalent, therefore provable. Then a proof of the two expressions with the letters $a_{\nu a.f}$ can be transformed back to a proof with the atoms $\nu a.f$ by simply substituting $\nu a.f$ for $a_{\nu a.f}$. However, note that it is not necessary to do the actual substitution; we can carry out the same proof on the original expressions with the $\nu a.f$.

For expressions of ν-depth greater than one, we simply perform the above construction inductively, innermost scopes first. We use the KA axioms and the semilattice identities on depth-n ν-subexpressions to determine the semilattice identities on depth-$(n-1)$ ν-subexpressions, then use the nominal axiom (5) and the rule (16) to prepare these semilattice identities for use on the next level.

This completes the proof.

4.5 More Detail

For an expression e, define $|e|_\nu$ to be the ν-height of e. More formally,

$$\begin{array}{lllll} |x|_\nu = 0 & |0|_\nu = 0 & |e_1 + e_2|_\nu = \max\{|e_1|_\nu, |e_2|_\nu\} & |e^*|_\nu = |e|_\nu \\ & |1|_\nu = 0 & |e_1 \cdot e_2|_\nu = \max\{|e_1|_\nu, |e_2|_\nu\} & |\nu x.e|_\nu = 1 + |e|_\nu \end{array}$$

If $|e|_\nu = 0$, then no ν appears in e; in other words, e is *ν-free*.

Proposition 4.3 *Let e_1, e_2 be arbitrary expressions. Then, $I(e_1) = I(e_2)$ implies that* $\mathsf{NKA} \vdash e_1 = e_2$.

Proof. The proof proceeds by induction on $\max\{|e_1|_\nu, |e_2|_\nu\}$. If this quantity is 0, then both e_1 and e_2 are ν-free regular expressions. In this case, the interpretations of $I(e_1)$ and $I(e_2)$ are the standard language interpretations, so by completeness of Kleene algebra, we have $\mathsf{KA} \vdash e_1 = e_2$, therefore $\mathsf{NKA} \vdash e_1 = e_2$.

If $\max\{|e_1|_\nu, |e_2|_\nu\} > 0$, let $k = \max\{|e_1|_\nu, |e_2|_\nu\} - 1$. According to the naming scheme of §4.3 for bound variables, there are subexpressions of e_1, e_2 at depth $k+1$ of the form $\nu x_k.f$, where f is a ν-free. Let $\nu x_k.f_1, \ldots, \nu x_k.f_m$ be an enumeration of all such subexpressions. Let $G_1, \ldots, G_N \subseteq \Sigma^*$ be the atoms of the Boolean algebra of subsets of Σ^* generated by the sets $I(f_1), \ldots, I(f_m)$. Since regular languages are closed under the set-theoretic Boolean operations, the sets $G_1, \ldots, G_N$ are regular, thus there exist regular expressions $g_1, \ldots, g_N$ such that $I(g_j) = G_j$, $1 \le j \le N$. Every set $I(f_i)$ can be written as the union of a set of atoms $\{G_j \mid j \in J_i\}$, where the index set $J_i \subseteq \{1, \ldots, N\}$. By completeness of KA, we have $\mathsf{KA} \vdash f_i = \sum_{j \in J_i} g_j$. By the axioms of nominal KA, we then have that $\mathsf{NKA} \vdash \nu x_k.f_i = \sum_{j \in J} \nu x_k.g_j$.

The construction of the previous paragraph allows us to rewrite the expressions e_1 and e_2 so that every subexpression starting with νx_k is of the form $\nu x_k.g_j$ for some $1 \le j \le N$.

Claim 4.4 Let h be an arbitrary expression and let $y_1, \ldots, y_N$ be distinct letters in Σ that do not appear in h. Let $h' = h[\nu x_k.g_j \mapsto y_j]_j$ be the expression that results by replacing every subterm $\nu x_k.g_j$ in h with y_j for $1 \le j \le N$. Then

(i) $h = h'[y_j \mapsto \nu x_k.g_j]_j$, and

(ii) $I(h') = I(h)[I(\nu x_k.g_j) \mapsto y_j]_j$.

Note that the substitution operation $[y_j \mapsto \nu x_k.g_j]_j$ allows capture.

Now take any collection $y_1, \ldots, y_N$ of distinct letters in Σ that do not appear in e_1 or e_2 and define $e_1' = e_1[\nu x_k.g_j \mapsto y_j]$ and $e_2' = e_2[\nu x_k.g_j \mapsto y_j]$. From the hypothesis $I(e_1) = I(e_2)$ and Claim 4.4(ii), we obtain that $I(e_1') = I(e_2')$. Since the ν-height of both e_1' and e_2' is strictly less than $k+1$, we can invoke the induction hypothesis to obtain that $\mathsf{NKA} \vdash e_1' = e_2'$. Using the substitution rule, we obtain

$$\mathsf{NKA} \vdash e_1'[y_j \mapsto \nu x_k.g_j]_j = e_2'[y_j \mapsto \nu x_k.g_j]_j,$$

hence $\mathsf{NKA} \vdash e_1 = e_2$ by virtue of Claim 4.4(i). □

5 Conclusion

We have presented results on completeness and incompleteness of nominal Kleene algebra as introduced by Gabbay and Ciancia [5]. There are various directions for future work.

We have not defined an automata-theoretic counterpart of nominal Kleene algebra. Having an adequate automaton model that recognizes the language denoted by an expression may open the door to a more efficient coalgebraic decision procedure, which would be of particular interest for the applications mentioned in the introduction. We note that the normalization procedure presented in this paper is highly impractical, as the preprocessing step of §4.1 involves an exponential blowup. However, coalgebraic decision procedures have been devised for KAT and NetKAT and have proven quite successful in applications, and we suspect that a similar approach may bear fruit here.

Another interesting direction would be to follow recent work by Joanna Ochremiak [7] involving nominal sets over atoms equipped with both relational and algebraic structure. This is an extension of the original work of Gabbay and Pitts, in which atoms can only be compared for equality.

The proof we have provided is very concrete and does not explore the rich categorical structure of nominal sets. It would be interesting to understand whether the proof can be phrased in more abstract terms, which would also be more amenable to generalizations such as those mentioned above.

Acknowledgments

We are grateful to Jamie Gabbay for bringing the original NKA paper to our attention. We would also like to thank Filippo Bonchi, Daniela Petrisan, Damien Pous, Paul Brunet, and Fabio Zanasi for various comments and suggestions.

References

[1] Mikolaj Bojanczyk, Bartek Klin, and Slawomir Lasota. Automata theory in nominal sets. *Logical Methods in Computer Science*, 10(3), 2014.

[2] Maribel Fernández and Murdoch J. Gabbay. Nominal rewriting with name generation: abstraction vs. locality. In *Proceedings of the 7th ACM SIGPLAN International Symposium on Principles and Practice of Declarative Programming (PPDP 2005)*. ACM Press, July 2005.

[3] Murdoch Gabbay and Andrew M. Pitts. A new approach to abstract syntax involving binders. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*, pages 214–224, 1999.

[4] Murdoch J. Gabbay. A study of substitution, using nominal techniques and Fraenkel-Mostowski sets. *Theoretical Computer Science*, 410(12-13), March 2009.

[5] Murdoch J. Gabbay and Vincenzo Ciancia. Freshness and name-restriction in sets of traces with names. In *Foundations of software science and computation structures, 14th International Conference (FOSSACS 2011)*, volume 6604 of *Lecture Notes in Computer Science*, pages 365–380. Springer, 2011.

[6] Murdoch J. Gabbay and Aad Mathijssen. Nominal universal algebra: equational logic with names and binding. *Journal of Logic and Computation*, 19(6):1455–1508, December 2009.

[7] Joanna Ochremiak. Nominal sets over algebraic atoms. In Peter Höfner, Peter Jipsen, Wolfram Kahl, and Martin Eric Müller, editors, *Relational and Algebraic Methods in Computer Science - 14th International Conference, RAMiCS 2014, Marienstatt, Germany, April 28-May 1, 2014. Proceedings*, volume 8428 of *Lecture Notes in Computer Science*, pages 429–445. Springer, 2014.