Locally Nameless Sets

ANDREW M. PITTS, University of Cambridge, UK

This paper provides a new mathematical foundation for the locally nameless representation of syntax with binders, one informed by nominal techniques. It gives an equational axiomatization of two key locally nameless operations, "variable opening" and "variable closing" and shows that a lot of the locally nameless infrastructure can be defined from that in a syntax-independent way, including crucially a "shift" functor for name binding. That functor operates on a category whose objects we call *locally nameless sets*. Functors combining shift with sums and products have initial algebras that recover the usual locally nameless representation of syntax with binders in the finitary case. We demonstrate this by uniformly constructing such an initial locally nameless set for each instance of Plotkin's notion of binding signature. We also show by example that the shift functor is useful for locally nameless sets of a semantic rather than a syntactic character. The category of locally nameless sets is proved to be isomorphic to a known topos of finitely supported M-sets, where M is the full transformation monoid on a countably infinite set. A corollary of the proof is that several categories that have been used in the literature to model variable renaming operations on syntax with binders are all equivalent to each other and to the category of locally nameless sets.

CCS Concepts: • **Theory of computation → Categorical semantics**; **Equational logic and rewriting**; *Logic and verification*; • **Software and its engineering → Syntax**.

Additional Key Words and Phrases: name binding, locally nameless, metatheory of syntax, cofinite quantification, category theory, initial algebra, Agda

ACM Reference Format:
Andrew M. Pitts. 2023. Locally Nameless Sets. *Proc. ACM Program. Lang.* 7, POPL, Article 17 (January 2023), 27 pages. https://doi.org/10.1145/3571210

1 INTRODUCTION

Very many programming languages and logics feature syntactical constructs for binding various sorts of name within lexical scopes. When reasoning about the properties of such languages, and especially when creating fully formalized and machine-checked proofs, it is practically essential to raise the level of abstraction when representing and computing with such syntax. Obviously abstract syntax trees are more convenient than strings of symbols when it comes to representing the structure of expressions. But if those syntax trees feature scoping constructs, it is extremely desirable to have a representation mechanism that automatically factors out renaming of bound names (α-equivalence). At the same time, the mechanisms for computing with representations and moving between such abstract representations and more concrete ones have to be expressive, machine implementable and humanly understandable. It is not easy to satisfy all these demands at once.

Over the years several mechanisms for representing binding modulo α-equivalence have been developed. One of the earliest was by de Bruijn [1972], who used numerical indices to count the number of binding operations passed through on the way from (or to) a name site. The higher-order

Author's address: Andrew M. Pitts, andrew.pitts@cl.cam.ac.uk, University of Cambridge, Department of Computer Science and Technology, 15 JJ Thomson Avenue, Cambridge, UK, CB3 0FD.

2475-1421/2023/1-ART17
https://doi.org/10.1145/3571210

approach of Pfenning and Elliott [1988] delegated the problem to be solved once and for all in a typed λ-calculus meta-framework, using one of the other approaches. At the other extreme Urban and Tasson [2005] retain explicit names and develop packages of proved theorems (and associated automation) for data quotiented by α-equivalence within classical higher-order logic by leveraging the nominal theory of Gabbay and Pitts [2002]. We refer the reader to the excellent survey of binding representations as of 2008 in Sect. 2 of the paper by Aydemir et al. [2008]; and to the blog post of Cockx [2021] for a recent survey with emphasis on dependently-typed functional programming using Agda [2023]. The various different approaches have different pros and cons when it comes to ease of use, ease of implementation, efficiency, etc.

After its survey, the paper of Aydemir et al. [2008] goes on to advocate the *locally nameless* approach to represent binding structure, combined with the use of *cofinite quantification*. A locally nameless representation combines a nominal approach to unbound names with the use of de Bruijn indices for bound ones. Thus free variables in terms still have names and those names do not change if we enlarge the context in which we are using the term; so one retains the practically very useful feature of the nominal approach that weakening (and equations between combinations of weakenings) is invisible. At the same time, through the use of deBruijn indices one gets canonical representation of syntax modulo α-equivalence as purely inductively defined sets without any need for use of quotients (particularly helpful for proof assistants based on type theories where inductive definition is the main mechanism for constructions). Given that free variables are still represented by names, there is no need to consider terms with "dangling" indices pointing to levels of binding higher than the ones that exist in the term. In other words one should cut down to the (still inductively defined) subset of "locally closed" terms. Doing so avoids one of the main problems with use of de Bruijn indices: the identity of a dangling index changes according to the context. For example if we substitute a term under a binding construct we may have to do some shifting on indices; and that can be error-prone for implementation and tricky for human understanding. The locally nameless approach avoids this pitfall by insisting that reasoning only takes place (in the end) for locally closed terms. The use of cofinite quantification over names is crucial for making this a viable approach. (This quantifier specialises to the distinctive freshness quantifier $\mathrel{\mathit{N}}$ of nominal logic [Pitts 2003] when the properties involved are finitely supported.) The material in Aydemir et al. [2008] and the follow-up journal paper by Charguéraud [2012] provide a compelling case that the locally nameless approach does a good job of satisfying the competing engineering demands involved in mechanisms for representing binding.

Here we address not so much the engineering aspects of the locally nameless approach, but rather its mathematical foundations. We abstract from existing concrete uses of the locally nameless representation a so-far unnoticed algebraic structure (the *opening/closing algebra* of Sect. 2.2) and show that it can be used to give a purely equational development of many of the key notions in the locally nameless approach (Sects 2 and 4). Why is this useful? For one thing, equational logic has proved very useful in computer science and algorithmic techniques for it are highly developed. Founding the locally nameless method on a relatively simple algebraic theory should facilitate development of logic and type theory designed to make it easier to deploy the locally nameless approach in practice (for example, by making invisible to the user some "boilerplate" aspects of the locally nameless method). However, there is a more immediately useful outcome: we are able to give an account of the locally nameless version of name binding (in the form of the *shift functor* of Sect. 3.4) that applies to *arbitrary* "locally nameless sets" (Definition 2.9) and not just ones that are inductively defined sets of finitary syntax; see Example 4.4 concerning semantic continuations. Before this work we only knew what locally nameless syntax means; now we know what "locally nameless semantics" means. This should be useful in practice, because it may enable

the pleasant properties of the locally nameless representation to be used for situations where syntax and semantics get mixed up – for example in proofs of normalization-by-evaluation.

The paper makes the following contributions:

- An equational axiomatization of *opening/closing operations* that suffices to derive, independently of any particular syntax, many of the other key concepts of the locally nameless representation of syntax with binders: abstraction of a name, concretion by a name and the "body" predicate; "not-a-free-variable-of" and "locally closed" properties, via a notion of *finite support*; and renaming and name swapping operations with expected properties. See Sect. 2.
- The algebras for our equational theory all of whose elements have finite support are called *locally nameless sets*. Using some existing semigroup theory, we prove that the category of locally nameless sets is isomorphic to a pre-existing category, namely the topos[1] of finitely supported M-sets where M is the *full transformation monoid* of all endofunctions on the countably infinite set of indices and names. In other words our equational axiomatization of opening/closing operations completely characterises the action on finitely supported objects of all functions mapping indices and names to indices or names. A corollary of the proof is that various categories for modelling *renaming* that have been considered in the literature [Gabbay and Hofmann 2008; Popescu 2022; Staton 2007] are all equivalent to each other and to this topos. See Sect. 3.
- We show that a *shift* endofunctor on locally nameless sets captures in a syntax-independent fashion the notion of name abstraction appropriate for binding constructs in the locally nameless representation. Combining it with sum and product functors, we prove that each of Plotkin's binding signatures automatically gives rise to an endofunctor of locally nameless sets whose initial algebra recovers the locally nameless representation of syntax over the signature with the correct notions of opening and closing, free variables and local closure. We also give an example (from denotational semantics) involving the shift functor applied to a non-syntactic locally nameless set. See Sect. 4.
- Agda [2023] was used to develop the theory of locally nameless sets and to check some of the proofs in the paper. The Agda code [Pitts 2023] mainly targets proofs that involve equational reasoning combined with the use of atoms and indices that are sufficiently fresh (via cofinite quantification). Some of these proofs involve a lot of nested case analysis on elements of sets with decidable equality (atoms and indices); some of our equational axioms are unfamiliar-looking and combinatorially complicated; and it is easy to forget to check necessary freshness conditions are satisfied when doing informal proofs. For all these reasons the use of an interactive theorem prover to produce machine-checked proofs was essential to gain assurance that the results in the paper are correct. Section 5 gives an overview of the Agda development.

2 DEFINITION OF LOCALLY NAMELESS SETS

The nominal sets model of syntax with binders begins with the observation that most syntactic constructions and properties are invariant under permuting names. So it considers abstract sets equipped with an action of such permutations. Any finite permutation is a composition of swapping operations and this leads to a presentation of nominal sets founded upon sets equipped with a *name swapping* operation satisfying a few simple equations; see [Pitts 2013, Section 6.1]. Here we take a similar approach, but replace name swapping with the two fundamental operations of the locally nameless representation: the "opening" operation that replaces a de Bruijn index with an atomic

[1] A *topos* is an elegant category theoretic notion that turns out to capture many aspects of (intuitionistic) higher-order logic and type theory; see for example [Johnstone 1977, 2002].

$$\{i \to a\}\{i \to b\}x = \{i \to b\}x \tag{OC$_1$}$$
$$\{i \leftarrow a\}\{j \leftarrow a\}x = \{j \leftarrow a\}x \tag{OC$_2$}$$
$$\{i \leftarrow a\}\{i \to a\}x = \{i \leftarrow a\}x \tag{OC$_3$}$$
$$\{i \to a\}\{i \leftarrow a\}x = \{i \to a\}x \tag{OC$_4$}$$
$$i \neq j \Rightarrow \{i \to a\}\{j \to b\}x = \{j \to b\}\{i \to a\}x \tag{OC$_5$}$$
$$a \neq b \Rightarrow \{i \leftarrow a\}\{j \leftarrow b\}x = \{j \leftarrow b\}\{i \leftarrow a\}x \tag{OC$_6$}$$
$$i \neq j \wedge a \neq b \Rightarrow \{i \to a\}\{j \leftarrow b\}x = \{j \leftarrow b\}\{i \to a\}x \tag{OC$_7$}$$
$$\{i \to b\}\{i \leftarrow a\}\{j \to b\}x = \{j \to b\}\{j \leftarrow a\}\{i \to a\}x \tag{OC$_8$}$$
$$\{j \leftarrow a\}\{i \to a\}\{j \leftarrow b\}x = \{j \leftarrow b\}\{i \to b\}\{i \leftarrow a\}x \tag{OC$_9$}$$

Fig. 1. The opening/closing axioms

name and the "closing" operation that abstracts away an atomic name by replacing it with an index. These operations have some simple equational properties (some of which are not so obvious) and this leads us to consider algebras for this equational theory, called opening/closing sets, or *OC-sets* for short. Omitted proofs can be found in the Agda development [Pitts 2023], as indicated.

2.1 Indices and Atoms [Pitts 2023, `Unfinite.agda`]

We take De Bruijn *indices* [de Bruijn 1972] to be elements of the set $\mathbb{N}$ of natural numbers (typical elements written $i, j, k, \ldots$). Indices are compared using the usual total order relation ($i < j$) on $\mathbb{N}$. *Atoms* are elements of a fixed set $\mathbb{A}$ (typical elements written $a, b, c, \ldots$) which we assume is disjoint from $\mathbb{N}$. Atoms are compared using equality ($a = b$) and its negation ($a \neq b$); in a constructive setting we need to assume that these are complementary, that is, $\mathbb{A}$ has decidable equality. Crucially, $\mathbb{A}$ is also assumed to be "unfinite"[2] in the sense that for each $n \in \mathbb{N}$

$$\forall a_1 \in \mathbb{A}, \ldots, \forall a_n \in \mathbb{A}, \exists a \in \mathbb{A},\ a \neq a_1 \wedge \cdots \wedge a \neq a_n \tag{1}$$

Since the set of natural numbers has these properties, one could take $\mathbb{A}$ to be a bijective copy of $\mathbb{N}$.

Definition 2.1 (Cofinite quantification). When discussing properties of atoms it is convenient to use the "for all but finitely many" quantifier. It is an important tool for reasoning about locally nameless representations of syntax with binders; see [Aydemir et al. 2008, Sect. 4]. To denote this quantifier we reuse the freshness quantifier symbol $\mathrel{\#}$ from nominal logic [Pitts 2003], since that is a special case of cofinite quantification restricted to predicates that are finitely supported in the sense of nominal sets. For any set X (with decidable equality, if working constructively) let $\mathrm{Fin}\, X$ denote the set of finite subsets of X. If $\Phi(x)$ is a property of elements $x \in X$, we write $\boxed{\mathrel{\#}x \in X,\ \Phi(x)}$ for the statement that $\Phi(x)$ *holds for all but finitely many* $x \in X$:

$$\mathrel{\#}x \in X,\ \Phi(x) \ \triangleq\ \exists A \in \mathrm{Fin}\, X, \forall x \in X,\ x \notin A \Rightarrow \Phi(x) \tag{2}$$

2.2 OC-Sets [Pitts 2023, `oc-Sets.agda`]

In the locally nameless representation of syntax involving binding operations [Aydemir et al. 2008; Charguéraud 2012] occurrences of free identifiers in terms retain their names, but occurrences of bound ones are anonymized using de Bruijn indices. Two fundamental operations on such terms are: "open" a term by replacing an index with a name; and "close" a term by replacing a name with an

[2]The terminology is due to André Joyal [private communication].

index. Abstracting away from syntactic details, given an arbitrary set X we define *opening/closing operations* for X to be given by a pair of functions $\mathbb{N} \times \mathbb{A} \times X \to X$, written

$$(i, a, x) \in \mathbb{N} \times \mathbb{A} \times X \;\mapsto\; \boxed{\{i \to a\}x} \in X \quad \text{“open index } i \text{ with atom } a \text{ in } x\text{”}$$
$$(i, a, x) \in \mathbb{N} \times \mathbb{A} \times X \;\mapsto\; \boxed{\{i \leftarrow a\}x} \in X \quad \text{“close atom } a \text{ with index } i \text{ in } x\text{”}$$

and satisfying the axioms in Fig. 1. An *oc-set* is a set equipped with such operations.

We shall see that sets of locally nameless terms equipped the “variable opening” and “variable closing” operations from [Charguéraud 2012, Sects 3.1 and 3.2] provide examples of oc-sets. A simple special case of that is provided by:

Example 2.2 (oc-set of indices and atoms). The following functions $\mathbb{N} \times \mathbb{A} \times X \to X$ with $X = \mathbb{N} \cup \mathbb{A}$ satisfy the opening/closing axioms. (Recall that $\mathbb{N}$ and $\mathbb{A}$ are assumed to be disjoint.)

$$\{i \to a\}j \triangleq \mathit{if}\ i = j\ \mathit{then}\ a\ \mathit{else}\ j \qquad \{i \to a\}b \triangleq b$$
$$\{i \leftarrow a\}j \triangleq j \qquad \{i \leftarrow a\}b \triangleq \mathit{if}\ a = b\ \mathit{then}\ i\ \mathit{else}\ b$$

Remark 2.3. The reader may well ask of Fig. 1, why those equations and why just those—are there some others we forgot? Axioms OC_1–OC_7 express straightforward commutation properties of “variable opening” and “variable closing” from [Charguéraud 2012], although they are not all explicitly stated there. In particular, axioms OC_3 and OC_4 imply that variable opening and closing are mutually inverse modulo suitable freshness and local closedness assumptions—the CLOSE_OPEN_VAR and OPEN_CLOSE_VAR properties on page 369 of *loc. cit.*; see Corollaries 2.5 and 2.8. The interesting point is that freshness and local closedness relations expressed in the abstract setting in terms of the open/closing operations (see Sects 2.3 and 2.4) coincide in concrete syntactic examples with their more usual definitions by induction on the structure of syntax; see Propositions 4.2 and 4.3. Axioms OC_8 and OC_9 are perhaps the least obvious ones. They allow us to express renaming (atom-to-atom) and re-indexing (index-to-index) functions in terms of opening and closing; see Sect. 2.7. Are the axioms enough? One pragmatic answer is that they suffice to develop the basic infrastructure of the locally nameless approach to representing syntax with binders independently of any particular inductively defined datatype of syntax. A more mathematical answer is provided by Theorem 3.5, which proves that the axioms in Fig. 1 completely characterise the action of endofunctions of $\mathbb{N} \cup \mathbb{A}$ on finitely supported objects.[3]

2.3 Freshness [Pitts 2023, `Freshness.agda`]

Let X be an oc-set. For each $a \in \mathbb{A}$ and $x \in X$, the *freshness* relation (cf. Remark 2.11 in [Pitts 2015]) is defined by

$$\boxed{a \# x} \;\triangleq\; \{0 \leftarrow a\}x = x \tag{3}$$

Lemma 2.4 below (which is an immediate consequence of axiom OC_2) shows that one can replace 0 in the above definition by any index without changing _ # _. We will see that in syntactic examples this relation coincides with the usual “a is not a free variable of x” relation (see Proposition 4.2).

Lemma 2.4. *Let X be an oc-set X. For all $i, j \in \mathbb{N}$, $a \in \mathbb{A}$ and $x \in X$*

$$\{i \leftarrow a\}x = x \;\Rightarrow\; \{j \leftarrow a\}x = x \tag{4}$$

Hence if $a \# x$ then for any $i \in \mathbb{N}$ we have $\{i \leftarrow a\}x = x$; and conversely if $\{i \leftarrow a\}x = x$ holds for some i, then $a \# x$. □

[3] There is also the question of independence; for example, the author believes that axioms OC_8 and OC_9 are not derivable from the others, but does not have a proof of that.

Combining the lemma with axiom OC_3 yields:

Corollary 2.5. *In any* OC*-set we have that* $a \# x \implies \{i \leftarrow a\}\{i \rightarrow a\}x = x$. □

2.4 Local Closedness [Pitts 2023, `LocalClosedness.agda`]

Let X be an OC-set. For each $i \in \mathbb{N}$ and $x \in X$ define

$$\boxed{i \succ x} \triangleq \forall j \geq i, \exists a \in \mathbb{A}, \ \{j \rightarrow a\}x = x \tag{5}$$

For syntactic examples the relation $i \succ x$ turns out to coincide with an inductively defined "x is closed_at_level_i" relation; see Proposition 4.3. In case $i = 0$ this is the local closure predicate, which plays a key role in the locally nameless approach to representation of syntax with binders; *cf.* Sect. 3.3 in [Charguéraud 2012]. For example, by restricting to locally closed elements one can avoid the need for error-prone index-shift operations when defining substitution.

The following is an immediate consequence of definition (5):

Lemma 2.6. *Let* X *be an* OC*-set* X*. For all* $i, j \in \mathbb{N}$*, and* $x \in X$*, if* $j \geq i$ *and* $i \succ x$*, then* $j \succ x$*. Hence if* $i \succ x$ *or* $j \succ x$*, then* $\max\{i, j\} \succ x$. □

Using OC_1 we have:

Lemma 2.7. *Let* X *be an* OC*-set. For all* $i \in \mathbb{N}$*,* $a, b \in \mathbb{A}$ *and* $x \in X$

$$\{i \rightarrow a\}x = x \implies \{i \rightarrow b\}x = x \tag{6}$$

Hence if $i \succ x$ *then for any* $j \geq i$ *and any* $a \in \mathbb{A}$ *we have* $\{j \rightarrow a\}x = x$. □

Combining this lemma with OC_4 yields:

Corollary 2.8. *In any* OC*-set we have that* $i \succ x \implies \{i \rightarrow a\}\{i \leftarrow a\}x = x$. □

2.5 Support [Pitts 2023, `Support.agda`]

Given an element $x \in X$ of an OC-set, say that $A \in \mathrm{Fin}\,\mathbb{A}$ *atom-supports* x if $\forall a \notin A, \ a \# x$ (so that A witnesses the fact that $\nexists a \in \mathbb{A}, a \# x$ holds). Say that $i \in \mathbb{N}$ *index-supports* x if $i \succ x$. We are interested in the case where such atom- and index-supports exist for each element.

Definition 2.9 (Locally nameless set). A *locally nameless set* is an OC-set X satisfying the following two properties:

$$\textit{finite atom-support}: \ \forall x \in X, \nexists a \in \mathbb{A}, \ a \# x \tag{7}$$

$$\textit{finite index-support}: \ \forall x \in X, \exists i \in \mathbb{N}, \ i \succ x \tag{8}$$

Example 2.10 (Locally nameless set of indices and atoms). Write $\boxed{\mathbb{N}\mathbb{A}}$ for the union $\mathbb{N} \cup \mathbb{A}$ of the disjoint sets of indices and of atoms. We saw in Example 2.2 how to make it into an OC-set. It is in fact a locally nameless set, because: each $a \in \mathbb{A}$ is atom-supported by $\{a\}$ and index-supported by 0; and each $i \in \mathbb{N}$ is atom-supported by $\emptyset$ and index-supported by $i + 1$.

Example 2.11 (Locally nameless set of λ-terms modulo α-equivalence). The locally nameless representation of all (open or closed) λ-terms modulo α-equivalence uses the inductively defined set Λ with constructors

$$\mathsf{bvar} : \mathbb{N} \to \Lambda, \quad \mathsf{fvar} : \mathbb{A} \to \Lambda, \quad \mathsf{lam} : \Lambda \to \Lambda, \quad \mathsf{app} : \Lambda \times \Lambda \to \Lambda \tag{9}$$

and with open/close operations recursively defined as follows (cf. sections 3.1 and 3.2 of [Charguéraud 2012]):

$$\{i \to a\}(\mathsf{bvar}\, j) \triangleq \mathit{if}\ i = j\ \mathit{then}\ \mathsf{fvar}\, a\ \mathit{else}\ \mathsf{bvar}\, j \qquad \{i \leftarrow a\}(\mathsf{bvar}\, j) \triangleq \mathsf{bvar}\, j$$
$$\{i \to a\}(\mathsf{fvar}\, b) \triangleq \mathsf{fvar}\, b \qquad \{i \leftarrow a\}(\mathsf{fvar}\, b) \triangleq \mathit{if}\ a = b\ \mathit{then}\ \mathsf{bvar}\, i\ \mathit{else}\ \mathsf{fvar}\, b$$
$$\{i \to a\}(\mathsf{lam}\, t) \triangleq \mathsf{lam}(\{i+1 \to a\}t) \qquad \{i \leftarrow a\}(\mathsf{lam}\, t) \triangleq \mathsf{lam}(\{i+1 \leftarrow a\}t)$$
$$\{i \to a\}(\mathsf{app}(t,t')) \triangleq \mathsf{app}(\{i \to a\}t, \{i \to a\}t') \qquad \{i \leftarrow a\}(\mathsf{app}(t,t')) \triangleq \mathsf{app}(\{i \leftarrow a\}t, \{i \leftarrow a\}t')$$

This definition satisfies the axioms in Fig. 1. Indeed it turns out that Λ is a locally nameless set that is the free algebra on $\mathbb{N}\mathbb{A}$ for a suitable endofunctor on the category that we introduce in Sect. 3. Not only this, but also the properties "not a free atom of" and "locally closed" that are defined inductively on the structure of terms and play a fundamental role in the locally nameless approach, coincide with the equationally defined notions of freshness and local closedness from sections 2.3 and 2.4; see the Agda development for details. We will see in Sect. 4 that all these properties of Λ are special cases of results that hold for the locally nameless representation of the syntax of terms modulo α-equivalence for any binding signature.

We will need the following two properties of the opening/closing operations with respect to freshness and local closedness. They follow from Lemma 2.4 together with axioms OC_1, OC_2, OC_5, OC_6 and OC_7; see the Agda development for the details.

Lemma 2.12. *Let X be an OC-set. For all $i \in \mathbb{N}$, $a \in \mathbb{A}$ and $x \in X$, if $A \in \mathrm{Fin}\,\mathbb{A}$ atom-supports x, then $A \cup \{a\}$ atom-supports $\{i \to a\}x$ and $A - \{a\}$ atom-supports $\{i \leftarrow a\}x$.* □

Lemma 2.13. *Let X be an OC-set. For all $i, j \in \mathbb{N}$, $a \in \mathbb{A}$ and $x \in X$*

$$j \succ x \Rightarrow j \succ \{i \to a\}x \tag{10}$$
$$i+1 \succ x \Rightarrow i \succ \{i \to a\}x \tag{11}$$
$$j \succ x \Rightarrow \max\{j, i+1\} \succ \{i \leftarrow a\}x \tag{12}$$

□

2.6 Abstraction and Concretion [Pitts 2023, AbstractionConcretion.agda]

As mentioned in the Introduction, in the locally nameless approach to representing syntax the meaningful part of a locally nameless set X is its *locally closed part*, defined as follows:

Definition 2.14. Given an OC-set X and $i \in \mathbb{N}$, define $\boxed{\mathrm{lc}_i(X)} \triangleq \{x \in X \mid i \succ x\}$. In case $i = 0$, $\mathrm{lc}_0(X)$ is called the *locally closed* part of X.

The $i = 0$ case of the opening and closing operations are of special interest. Adopting nominal terminology but using the notation from [Aydemir et al. 2008; Charguéraud 2012], given a locally nameless set X, for all $x \in X$ and $a \in \mathbb{A}$ we define

$$\begin{array}{rcll} \boxed{\backslash^a x} & \triangleq & \{0 \leftarrow a\}x & \text{"abstract atom } a \text{ in } x\text{"} \\ \boxed{x^a} & \triangleq & \{0 \to a\}x & \text{"concrete } x \text{ at atom } a\text{"} \end{array} \tag{13}$$

From (11) and (12) we have

$$x \in \mathrm{lc}_1(X) \Rightarrow x^a \in \mathrm{lc}_0(X) \tag{14}$$
$$x \in \mathrm{lc}_0(X) \Rightarrow {}^{\backslash a}x \in \mathrm{lc}_1(X) \tag{15}$$

By Corollaries 2.5 and 2.8 we also have

$$a \# x \;\Rightarrow\; {}^{\backslash a}(x^a) = x \tag{16}$$

$$x \in \mathrm{lc}_0(X) \;\Rightarrow\; ({}^{\backslash a}x)^a = x \tag{17}$$

(*cf.* properties CLOSE_OPEN_VAR and OPEN_CLOSE_VAR from [Charguéraud 2012, p. 369]).

Remark 2.15 (body). Charguéraud [2012] defines a "body t" predicate for asserting that a term t describes the body of a locally closed abstraction: for all but finitely many names a, the concretion of t at a should be locally closed. For elements x of a locally nameless set X this definition becomes: $\mathrm{body}\, x \triangleq \forall a \in \mathbb{A}.\ x^a \in \mathrm{lc}_0(X)$. However, this turns out to be equivalent to x being in $\mathrm{lc}_1(X)$:

$$\mathrm{body}\, x \;\Leftrightarrow\; 1 \succ x \tag{18}$$

The right-to-left implication in (18) is immediate from (14). In the other direction, if $A \in \mathrm{Fin}\,\mathbb{A}$ witnesses that $\forall a \in \mathbb{A}.\ x^a \in \mathrm{lc}_0(X)$ holds, then since x has finite atom-support we can choose some $a \notin A$ with $a \# x$. So $x^a \in \mathrm{lc}_0(X)$ (by assumption, since $a \notin A$) and hence by (15), we have ${}^{\backslash a}(x^a) \in \mathrm{lc}_1(X)$; but since $a \# x$, (16) gives us $x \in \mathrm{lc}_1(X)$, as required.

2.7 Re-naming, Re-indexing and Name Swapping

[Pitts 2023, `RenamingReindexingSwapping.agda`]

Using the opening/closing operations of a locally nameless set it is possible to define operations for atom-for-atom substitution and atom swapping. To see this we need the following lemma whose proof uses axioms OC_8 and OC_9 for the first time (together with Lemmas 2.4 and 2.7); the details are in the Agda development.

Lemma 2.16. *Let X be a locally nameless set. For all $a, b \in \mathbb{A}$, $i, j \in \mathbb{N}$ and $x \in X$*

$$i \succ x \;\wedge\; j \succ x \;\Rightarrow\; \{i \to b\}\{i \leftarrow a\}x = \{j \to b\}\{j \leftarrow a\}x \tag{19}$$

$$a \# x \;\wedge\; b \# x \;\Rightarrow\; \{j \leftarrow a\}\{i \to a\}x = \{j \leftarrow b\}\{i \to b\}x \tag{20}$$

In view of the this lemma, we get well-defined operations of *renaming* and *re-indexing* for a locally nameless set:

$$\boxed{[b \leftarrow\!\!\shortmid\, a]x} \;\triangleq\; \{i \to b\}\{i \leftarrow a\}x \quad \text{for some/any } i \succ x \tag{21}$$

$$\boxed{[i \mapsto j]x} \;\triangleq\; \{j \leftarrow a\}\{i \to a\}x \quad \text{for some/any } a \# x \tag{22}$$

Proposition 2.17 (Rensets). *The renaming operation (21) satisfies the axioms for* rensets *given by Popescu [2022]:*

$$[a \leftarrow\!\!\shortmid\, a]x = x \tag{23}$$

$$a \neq c \Rightarrow [b \leftarrow\!\!\shortmid\, a][c \leftarrow\!\!\shortmid\, a]x = [c \leftarrow\!\!\shortmid\, a]x \tag{24}$$

$$[c \leftarrow\!\!\shortmid\, b][b \leftarrow\!\!\shortmid\, a]x = [c \leftarrow\!\!\shortmid\, a][c \leftarrow\!\!\shortmid\, b]x \tag{25}$$

$$b \neq a' \neq a \neq b' \Rightarrow [b \leftarrow\!\!\shortmid\, a][b' \leftarrow\!\!\shortmid\, a']x = [b' \leftarrow\!\!\shortmid\, a'][b \leftarrow\!\!\shortmid\, a]x \tag{26}$$

(Popescu uses a slightly more complicated axiom than (25), namely

$$b \neq d \Rightarrow [c \leftarrow\!\!\shortmid\, b][b \leftarrow\!\!\shortmid\, a][d \leftarrow\!\!\shortmid\, b]x = [c \leftarrow\!\!\shortmid\, a][d \leftarrow\!\!\shortmid\, b]x \tag{27}$$

but (27) is implied by (25) modulo the other axioms, and use of the latter gives an equivalent category of finitely supported rensets.) *Thus every OC-set is one of Popescu's rensets.*

Proof. (23) follows from OC_4 and Lemma 2.7; (24) follows from OC_1 and Lemma 2.12; (25) follows from OC_8 and OC_9; and (26) follows from OC_5, OC_6 and OC_7. □

Lemma 2.18. *Given a locally nameless set X, for any $x \in X$ and $a \in \mathbb{A}$ we have*

$$a \# x \;\Rightarrow\; (\forall b \in \mathbb{A}, [b \leftarrow\!\!\shortmid a]x = x) \;\Rightarrow\; (\nu b \in \mathbb{A}, [b \leftarrow\!\!\shortmid a]x = x) \;\Rightarrow\; a \# x \tag{28}$$

Proof. The first implication follows from definition (21), the second is trivial and the third follows from (21) and OC_9. □

Corollary 2.19. *Given a locally nameless set X, for any $x \in X$ and $a, b, c \in \mathbb{A}$, if $c \# x$ and $c \neq b$, then $c \# [b \leftarrow\!\!\shortmid a]x$.*

Proof. In case $b = a$ the result follows from (23). If $b \neq a$, then it follows from the lemma using (24) and (26). □

Remark 2.20 (Locally nameless sets as nominal sets). Property (28) implies that x satisfies $\nu a \in \mathbb{A}, \nu b \in \mathbb{A}, [b \leftarrow\!\!\shortmid a]x = x$, which means that x is finitely supported in the sense of rensets [Popescu 2022, Sect. 4]. Thus every locally nameless set is naturally a finitely supported renset. Popescu [2022] already notes that every finitely supported renset is also a nominal set, using the fact that nominal sets can be presented in terms of a *name swapping operation* $x \mapsto (a\ b)x$ [Pitts 2013, Sect. 6.1]. Name swapping can be defined in terms of renaming in the usual way (using some/any fresh third name as an intermediary):

$$\boxed{(a\ b)x} \;\triangleq\; [b \leftarrow\!\!\shortmid c][a \leftarrow\!\!\shortmid b][c \leftarrow\!\!\shortmid a]x \quad \text{for some/any } c \text{ with } c \# x \text{ and } c \neq a, b \tag{29}$$

That this definition is independent of the choice of c follows from Proposition 2.17, Lemma 2.18 and Corollary 2.19. The definition works for any renset, but for locally nameless sets one can show that it is equivalent to the following definition in terms of opening and closing

$$(a\ b)x \;=\; \{j \to a\}\{i \to b\}\{j \leftarrow b\}\{i \leftarrow a\}x \quad \text{for some/any } i, j > x \text{ with } i \neq j \tag{30}$$

As for rensets, the nominal sets notion of support agrees with that of finite atom-support from Definition 2.9 when we regard a locally nameless set as a nominal set as above. Although not every nominal set is a locally nameless set, we will see in Sect. 3.5 that it is possible to freely generate a locally nameless set from a nominal set.

We need the following properties of renaming and swapping for Theorem 3.5:

Lemma 2.21. *In any locally nameless set the name-swapping (29) and renaming operations (21) satisfy:*

$$(a\ a)x = x \tag{31}$$

$$(a\ b)(a\ b)x = x = (a\ b)(b\ a)x \tag{32}$$

$$b \neq c \neq a \neq d \neq b \;\Rightarrow\; (a\ b)(c\ d)x = (c\ d)(a\ b)x \tag{33}$$

$$b \neq c \neq a \;\Rightarrow\; (a\ b)(c\ a)x = (c\ b)(a\ b)x \tag{34}$$

$$b \neq c \Rightarrow [c \leftarrow\!\!\shortmid b](a\ b)x = [a \leftarrow\!\!\shortmid b][c \leftarrow\!\!\shortmid a]x \tag{35}$$

$$b \neq c \neq a \neq d \neq b \;\Rightarrow\; (a\ b)[d \leftarrow\!\!\shortmid c]x = [d \leftarrow\!\!\shortmid c](a\ b)x \tag{36}$$

$$b \neq c \neq a \;\Rightarrow\; (a\ b)[a \leftarrow\!\!\shortmid c]x = [b \leftarrow\!\!\shortmid c](a\ b)x \tag{37}$$

$$b \neq c \neq a \;\Rightarrow\; (a\ b)[c \leftarrow\!\!\shortmid a]x = [c \leftarrow\!\!\shortmid b](a\ b)x \tag{38}$$

$$(a\ b)[b \leftarrow\!\!\shortmid a]x = [a \leftarrow\!\!\shortmid b](a\ b)x \tag{39}$$

Proof. These properties follow from Proposition 2.17 by calculations similar to the ones in the proof of that proposition. The details can be found in the accompanying Agda development. □

Remark 2.22 (Duality of opening and closing). Although indices and atoms play different roles when it comes to the notion of support (Sect. 2.5) and to modelling binding operations (see Sect. 4), there is a symmetry between them at the level of the equational theory in Fig. 1. Given an equation derived from the axioms in that figure, suppose the equation involves indices and atoms from among the distinct lists $i_1, \ldots, i_n$ and $a_1, \ldots, a_n$ for some n. Consider the transformation that interchanges each i_k with a_k at the same time interchanging opening operations $\{\_ \to \_\}$ with closing operations $\{\_ \leftarrow \_\}$. Under this transformation OC_1 corresponds with OC_2, OC_3 with OC_4, OC_5 with OC_6, OC_7 with itself, and OC_8 with OC_9. Thus the original equation is transformed into another valid equation.

3 THE CATEGORY OF LOCALLY NAMELESS SETS

In this section we define a category of locally nameless sets and prove that it is isomorphic to a pre-existing topos, namely the topos of finitely supported M-sets where M is the *full transformation monoid* of all functions $\mathbb{N} \cup \mathbb{A} \to \mathbb{N} \cup \mathbb{A}$ (Theorem 3.5). In particular this shows that the axioms in Fig. 1 completely characterise the action of endofunctions of $\mathbb{N} \cup \mathbb{A}$ on finitely supported objects. We also define an endofunctor on the category of locally nameless sets that captures in a syntax-independent fashion the notion of name abstraction appropriate for binding constructs in the locally nameless representation.

3.1 The Category of OC-Sets [Pitts 2023, `Category.agda`]

Recall the definition of *OC-set* from Sect. 2.2. A morphism $f : X \to Y$ of OC-sets is by definition a function from the set X to the set Y that commutes with the opening and closing operations:

$$f(\{i \to a\}x) = \{i \to a\}(f\,x) \qquad f(\{i \leftarrow a\}x) = \{i \leftarrow a\}(f\,x) \tag{40}$$

Clearly the collection of such functions is closed under composition and contains identity functions. So we get a category of OC-sets and morphisms, that we denote by $\boxed{\text{OC-}\mathbf{Set}}$.

Remark 3.1. Up to isomorphism OC-**Set** is the category $\mathbf{Set}^{OC}$ of (left) OC-sets for the monoid OC freely generated by the symbols $o_{(i,a)}$ and $c_{(i,a)}$ (as (i, a) ranges over $\mathbb{N} \times \mathbb{A}$) quotiented by the monoid congruence generated by equations corresponding to the axioms in Fig. 1; for example, axiom OC_1 corresponds to the equation $o_{(i,a)}o_{(i,b)} = o_{(i,b)}$, and similarly for the other axioms.

To see that there is an isomorphism between the categories OC-**Set** and $\mathbf{Set}^{OC}$, recall that left actions of any monoid M on a set X correspond to monoid morphisms from M into the full transformation monoid T_X whose elements are all endofunctions of X with the monoid operation being function composition and the unit being the identity function on X. So when M is the monoid OC defined above, actions of it on X correspond to functions mapping the generators $o_{(i,a)}$ and $c_{(i,a)}$ to functions $X \to X$ that satisfy the properties in Fig. 1, in other words, correspond to opening and closing operations that make X into an OC-set. Furthermore, functions that preserve OC-actions are the same thing as functions that commute with the opening and closing operations. Thus OC-**Set** is isomorphic to the category $\mathbf{Set}^{OC}$. (The notation $\mathbf{Set}^{OC}$ is justified by the fact that this is a presheaf category, namely the category of **Set**-valued functors and natural transformations from OC regarded as a category with one object and morphisms given by the elements of OC.)

Lemma 3.2. *Let $f : X \to Y$ be a morphism of OC-sets. Then for all $a \in \mathbb{A}$, $i \in \mathbb{N}$ and $x \in X$*

$$a \# x \Rightarrow a \# f\,x \tag{41}$$

$$i > x \Rightarrow i > f\,x \tag{42}$$

Hence if x is finitely atom-supported (respectively finitely index-supported) in X, then $f\,x$ is finitely atom-supported (respectively finitely index-supported) in Y.

Proof. From (40) we get that $\{0 \leftarrow a\}x = x$ implies $\{0 \leftarrow a\}(f\,x) = f(\{0 \leftarrow a\}x) = f\,x$, and $\{i \to a\}x = x$ implies $\{i \to a\}(f\,x) = f(\{i \to a\}x) = f\,x$. So if $A \in \mathrm{Fin}\,\mathbb{A}$ atom-supports x in X, then it also atom supports $f\,x$ in Y; and if i index-supports x in X it also index-supports $f\,x$ in Y. □

As for any category of M-sets, the product $X \times Y$ of two objects X and Y in the category $\mathbf{Set}^{OC}$ is given by the cartesian product of their underlying sets equipped with the OC-action: $m \cdot (x, y) = (m \cdot x, m \cdot y)$. So in particular the opening/closing operations for the product of two oc-sets are given by openening/closing in each component. From this we immediately have:

Lemma 3.3. *Given oc-sets X and Y, if $a \in \mathbb{A}$, $i \in \mathbb{N}$, $x \in X$ and $y \in Y$, then in $X \times Y$ we have:*

$$a \# (x, y) \Leftrightarrow a \# x \wedge a \# y \tag{43}$$

$$i \succ (x, y) \Leftrightarrow i \succ x \wedge i \succ y \tag{44}$$

In particular, if $A \in \mathrm{Fin}\,\mathbb{A}$ atom-supports x in X and $B \in \mathrm{Fin}\,\mathbb{A}$ atom-supports y in Y, then $A \cup B$ atom-supports (x, y) in $X \times Y$; and if $i \in \mathbb{N}$ index-supports x in X and j index-supports y in Y, then $\max\{i, j\}$ *index-supports (x, y) in $X \times Y$.* □

3.2 The Category Lns

Definition 3.4. Given a set S, let $\boxed{\mathrm{T}_S}$ denote the *full transformation semigroup* on S. This is the set of all functions $S \to S$ equipped with the associative operation of function composition. T_S is not just a semigroup, but also a monoid since it has a unit element given by the identity function on S. As for any monoid, regarding it as a one-object category we can consider the presheaf category $\boxed{\mathbf{Set}^{\mathrm{T}_S}}$ of **Set**-valued functors and natural transformations on T_S. More concretely its objects are T_S-sets, that is, sets X equipped with a unitary and associative action $\_ \cdot \_ : \mathrm{T}_S \times X \to X$; and its morphisms are functions that preserve the action. Given an object $(X, \cdot)$ in $\mathbf{Set}^{\mathrm{T}_S}$ one says that $x \in X$ is *supported* by $A \in \mathrm{Fin}\,S$ if

$$\forall m, m' \in \mathrm{T}_S,\ (\forall a \in A,\ m(a) = m'(a)) \implies m \cdot x = m' \cdot x \tag{45}$$

$(X, \cdot)$ is said to be *finitely supported* if for all $x \in X$ there exists an $A \in \mathrm{Fin}\,S$ that supports x. Let $\boxed{(\mathbf{Set}^{\mathrm{T}_S})_{\mathrm{fs}}}$ denote the full subcategory of $\mathbf{Set}^{\mathrm{T}_S}$ whose objects are the finitely supported T_S-sets.

It is not hard to see that $(\mathbf{Set}^{\mathrm{T}_S})_{\mathrm{fs}}$ is closed under taking finite limits in the category $\mathbf{Set}^{\mathrm{T}_S}$ and that the inclusion $(\mathbf{Set}^{\mathrm{T}_S})_{\mathrm{fs}} \hookrightarrow \mathbf{Set}^{\mathrm{T}_S}$ has a right adjoint given on objects by sending $(X, \cdot) \in \mathbf{Set}^{\mathrm{T}_S}$ to $(X_{\mathrm{fs}}, \cdot)$, where $\boxed{X_{\mathrm{fs}}}$ is the subset of X consisting of finitely supported elements. It is easy to see that the inclusion is comonadic and therefore we can apply [Johnstone 1977, Theorem 2.32] to conclude that $(\mathbf{Set}^{\mathrm{T}_S})_{\mathrm{fs}}$ is a topos with a geometric surjection to it from the presheaf topos $\mathbf{Set}^{\mathrm{T}_S}$.

Theorem 3.5. *Let $\boxed{\mathbf{Lns}}$ denote the full subcategory of oc-**Set** whose objects are the locally nameless sets (Sect. 2.5). Then* **Lns** *is isomorphic to the topos $(\mathbf{Set}^{\mathrm{T}_S})_{\mathrm{fs}}$ when $S = \mathbb{N} + \mathbb{A}$, the union of the set $\mathbb{N}$ of indices and the set $\mathbb{A}$ of atoms.*

The proof of this theorem occupies the next section. The theorem implies that **Lns** has rich categorical properties. For example it is cartesian closed and has a subobject classifier (and can interpret intuitionistic higher-order logic and dependent type theory). That structure can be described by transferring across the isomorphism $(\mathbf{Set}^{\mathrm{T}_{\mathbb{N}+\mathbb{A}}})_{\mathrm{fs}} \cong \mathbf{Lns}$ known descriptions of exponentials and subobject classifiers in toposes of finitely supported M-sets (*cf.* [Pitts 2015]). Those descriptions have a Kripke-like character typical of presheaf toposes that make them more complicated to use than, for example, the topos structure of nominal sets [Pitts 2013]. However, some aspects of the category **Lns** are simple. For example, it is not hard to see that the forgetful functor $\mathbf{Lns} \to \mathbf{Set}$ creates finite limits and filtered colimits. We will rely on this fact implicitly in Sects 3.5 and 4.

$$\tau_{a,a} = \mathrm{id} \tag{TS$_1$}$$
$$\tau_{b,a} \circ \tau_{b,a} = \mathrm{id} = \tau_{b,a} \circ \tau_{a,b} \tag{TS$_2$}$$
$$b \neq c \neq a \neq d \neq b \;\Rightarrow\; \tau_{b,a} \circ \tau_{d,c} = \tau_{d,c} \circ \tau_{b,a} \tag{TS$_3$}$$
$$b \neq c \neq a \;\Rightarrow\; \tau_{b,a} \circ \tau_{a,c} = \tau_{b,c} \circ \tau_{b,a} \tag{TS$_4$}$$
$$\varepsilon_{a,a} = \mathrm{id} \tag{TS$_5$}$$
$$c \neq a \;\Rightarrow\; \varepsilon_{b,a} \circ \varepsilon_{c,a} = \varepsilon_{c,a} \tag{TS$_6$}$$
$$\varepsilon_{c,b} \circ \varepsilon_{b,a} = \varepsilon_{c,a} \circ \varepsilon_{c,b} \tag{TS$_7$}$$
$$b \neq c \neq a \neq d \;\Rightarrow\; \varepsilon_{d,c} \circ \varepsilon_{b,a} = \varepsilon_{b,a} \circ \varepsilon_{d,c} \tag{TS$_8$}$$
$$c \neq b \;\Rightarrow\; \varepsilon_{c,b} \circ \tau_{b,a} = \varepsilon_{a,b} \circ \varepsilon_{c,a} \tag{TS$_9$}$$
$$b \neq c \neq a \neq d \neq b \;\Rightarrow\; \tau_{b,a} \circ \varepsilon_{d,c} = \varepsilon_{d,c} \circ \tau_{b,a} \tag{TS$_{10}$}$$
$$b \neq c \neq a \;\Rightarrow\; \tau_{b,a} \circ \varepsilon_{a,c} = \varepsilon_{b,c} \circ \tau_{b,a} \tag{TS$_{11}$}$$
$$b \neq c \neq a \;\Rightarrow\; \tau_{b,a} \circ \varepsilon_{c,a} = \varepsilon_{c,b} \circ \tau_{b,a} \tag{TS$_{12}$}$$
$$\tau_{b,a} \circ \varepsilon_{b,a} = \varepsilon_{a,b} \circ \tau_{b,a} \tag{TS$_{13}$}$$

Fig. 2. Axioms for full transformation semigroups

Remark 3.6 (Equivalent categories of renaming sets). A corollary of the proof in Sect. 3.3 is that $(\mathbf{Set}^{\mathrm{T}_{\mathbb{A}}})_{\mathrm{fs}}$ is isomorphic to the category of *finitely supported rensets* introduced by Popescu [2022]. The proof also shows that cutting down from the full transformation monoid $\mathrm{T}_{\mathbb{A}}$ to the submonoid $(\mathrm{T}_{\mathbb{A}})_{\mathrm{fin}}$ of finite endofunctions, we also have that $(\mathbf{Set}^{\mathrm{T}_{\mathbb{A}}})_{\mathrm{fs}}$ is isomorphic to $(\mathbf{Set}^{(\mathrm{T}_{\mathbb{A}})_{\mathrm{fin}}})_{\mathrm{fs}}$. The latter is the category of *nominal renaming sets* studied by Gabbay and Hofmann [2008] and proved equivalent by them to the category used by Staton [2007]; this confirms a conjecture of Popescu [2022, p 634]. Assuming $\mathbb{A}$ is in bijection with $\mathbb{N}$ (so that $\mathbb{N}\mathbb{A} \cong \mathbb{A}$), then $(\mathbf{Set}^{\mathrm{T}_{\mathbb{A}}})_{\mathrm{fs}}$ is isomorphic to $(\mathbf{Set}^{\mathrm{T}_{\mathbb{N}\mathbb{A}}})_{\mathrm{fs}}$ and so by the theorem, all of these categories are equivalent to **Lns**.

3.3 Proof of Theorem 3.5

For any set S the full transformation semigroup T_S (Definition 3.4) contains elements $\varepsilon_{b,a}$ and $\tau_{b,a}$, where for all $a, b, c \in S$

$$\boxed{\varepsilon_{b,a}(c)} \triangleq \mathit{if}\ c = a\ \mathit{then}\ b\ \mathit{else}\ c \tag{46}$$
$$\boxed{\tau_{b,a}(c)} \triangleq \mathit{if}\ c = a\ \mathit{then}\ b\ \mathit{else\ if}\ c = b\ \mathit{then}\ a\ \mathit{else}\ c \tag{47}$$

Let $\boxed{(\mathrm{T}_S)_{\mathrm{fin}}}$ denote the submonoid of the full transformation monoid T_S whose elements are the functions $m : S \to S$ that are *finite* in the sense that $\forall a \in S,\ m(a) = a$ holds, or equivalently $[m{\neq}\mathrm{id}]$ is a finite set, where in general we write

$$\boxed{[m{\neq}m']} \triangleq \{a \in S \mid m(a) \neq m'(a)\} \qquad (m, m' : S \to S) \tag{48}$$

Note that each $a, b \in S$ we have $\varepsilon_{b,a}, \tau_{b,a} \in (\mathrm{T}_S)_{\mathrm{fin}}$, because $[\varepsilon_{b,a}{\neq}\mathrm{id}] \subseteq \{a\}$ and $[\tau_{b,a}{\neq}\mathrm{id}] \subseteq \{a, b\}$. Presentations of full transformation monoids on *finite* sets are known in the literature on semigroups, beginning with Iwahori and Iwahori [1974]. Here we use Ganyushkin and Mazorchuk [2009] to deduce the following result about the monoid $(\mathrm{T}_S)_{\mathrm{fin}}$ for any, possibly infinite, set S.

Proposition 3.7. *The monoid $(\mathrm{T}_S)_{\mathrm{fin}}$ is freely generated by the elements $\varepsilon_{b,a}$ and $\tau_{b,a}$ (where $a, b \in S$) subject to the equations in Fig. 2.*

Proof. For each $A \in \mathrm{Fin}\, S$ consider the monoid generated by symbols $\varepsilon_{b,a}$ and $\tau_{b,a}$ with $a, b \in A$, subject to the equations in Fig. 2. Equations TS_1–TS_4 are known to present the symmetric group of finite permutations; see for example [Pitts 2013, Sect. 6.1] (which replaces TS_3 and TS_4 with the single equation $\tau_{b,a} \circ \tau_{d,c} \circ \tau_{b,a} = \tau_{\tau_{b,a}(d),\tau_{b,a}(c)}$, which is equivalent to them modulo TS_1 and TS_2). Therefore it follows from Ganyushkin and Mazorchuk [2009, Theorem 9.3.1] that the whole set of equations TS_1–TS_{13} restricted to $a, b, c, d \in A$ presents the finite full transformation semigroup T_A with symbol $\varepsilon_{b,a}$ (respectively $\tau_{b,a}$) standing for the function $A \to A$ given by (46) (respectively (47)). We can identify T_A with the submonoid of $(\mathrm{T}_S)_{\mathrm{fin}}$ given by the finite functions $m \in \mathrm{T}_S$ with $[m{\neq}\mathrm{id}] \subseteq A$; and under this identification the generators of T_A become the finite endofunctions on S defined in (46) and (47). Since every $m \in (\mathrm{T}_S)_{\mathrm{fin}}$ is in some T_A (for example $A = [m{\neq}\mathrm{id}]$) it follows that these functions freely generate the whole of $(\mathrm{T}_S)_{\mathrm{fin}}$ subject to the equations in Fig. 2. □

Now in Definition 3.4 taking S to be the union $\mathbb{N}\mathbb{A}\backslash$ of $\mathbb{N}$ and $\mathbb{A}\backslash$, we show that there is a functor $F : (\mathbf{Set}^{\mathrm{T}_{\mathbb{N}\mathbb{A}\backslash}})_{\mathrm{fs}} \to \mathbf{Lns}$ sending each $(X, \cdot) \in (\mathbf{Set}^{\mathrm{T}_{\mathbb{N}\mathbb{A}\backslash}})_{\mathrm{fs}}$ to the locally nameless set with underlying set X and opening/closing operations given for all $i \in \mathbb{N}$ and $a \in \mathbb{A}\backslash$ by:

$$\{i \to a\}x \triangleq \varepsilon_{a,i} \cdot x \qquad\qquad \{i \leftarrow a\}x \triangleq \varepsilon_{i,a} \cdot x \tag{49}$$

These operations satisfy the axioms in Fig. 1 because of the oc-set structure of $\mathbb{N}\mathbb{A}\backslash$ (Example 2.2). That each $x \in X$ has finite atom-support (7) and finite index-support (8) follows from the fact that $(X, \cdot)$ is finitely supported, so that there is some $A \in \mathrm{Fin}\, \mathbb{N}\mathbb{A}\backslash$ satisfying (45). For then if $a \in \mathbb{A}\backslash - A$, we have $\forall c \in A,\ \varepsilon_{0,a}(c) = c = \mathrm{id}(c)$ so that by (45) we have $\{0 \leftarrow a\}x = \varepsilon_{0,a} \cdot x = \mathrm{id} \cdot x = x$ and thus $a \# x$; similarly if $i \in \mathbb{N}$ satisfies $i \geq 1{+}\max(\mathbb{N} \cap A)$, then for any $\forall a \in \mathbb{A}\backslash, \forall c \in A,\ \varepsilon_{a,i}(c) = c = \mathrm{id}(c)$ so that $\{i \to a\}x = \varepsilon_{a,i} \cdot x = \mathrm{id} \cdot x = x$ and thus $1 + \max(\mathbb{N} \cap A) \succ x$.

We can take the action of $F : (\mathbf{Set}^{\mathrm{T}_{\mathbb{N}\mathbb{A}\backslash}})_{\mathrm{fs}} \to \mathbf{Lns}$ on morphisms to be the identity, because if $f : (X, \cdot) \to (Y, \cdot)$ is a morphism in $(\mathbf{Set}^{\mathrm{T}_{\mathbb{N}\mathbb{A}\backslash}})_{\mathrm{fs}}$ then in particular f commutes with the opening and closing operations defined by (49). Trivially this makes F functorial (a faithful functor, in fact). *To prove the theorem we will show that $F : (\mathbf{Set}^{\mathrm{T}_{\mathbb{N}\mathbb{A}\backslash}})_{\mathrm{fs}} \to \mathbf{Lns}$ is an isomorphism.*

Proposition 3.8. *For each locally nameless set X there is a unique unitary associative action $\_ \odot \_ : (\mathrm{T}_{\mathbb{N}\mathbb{A}\backslash})_{\mathrm{fin}} \times X \to X$ satisfying for all $a \in \mathbb{A}\backslash$ and $i \in \mathbb{N}$*

$$\varepsilon_{a,i} \odot x = \{i \to a\}x \qquad\qquad \varepsilon_{i,a} \odot x = \{i \leftarrow a\}x \tag{50}$$

Furthermore if $f : X \to Y$ in **Lns**, *then for all $m \in (\mathrm{T}_{\mathbb{N}\mathbb{A}\backslash})_{\mathrm{fin}}$*

$$\forall x \in X,\ f(m \odot x) = m \odot (f\, x) \tag{51}$$

Proof. Recall that unitary associative actions of $(\mathrm{T}_{\mathbb{N}\mathbb{A}\backslash})_{\mathrm{fin}}$ on a set X correspond to monoid morphisms $(\mathrm{T}_{\mathbb{N}\mathbb{A}\backslash})_{\mathrm{fin}} \to \mathrm{T}_X$ and hence by Proposition 3.7 to functions mapping the generators to elements of T_X satisfying the equations in Fig. 2. Since $\mathbb{N}\mathbb{A}\backslash$ is divided into two halves, on the face of it[4] there are eight different kinds of generator: $\varepsilon_{a,i}$, $\varepsilon_{i,a}$, $\varepsilon_{b,a}$, $\varepsilon_{j,i}$, $\tau_{b,a}$, $\tau_{j,i}$, $\tau_{a,i}$ and $\tau_{i,a}$, where $i, j \in \mathbb{N}$ and $a, b \in \mathbb{A}\backslash$. If X is a locally nameless set and we map the first two kinds of generator to the functions in (50), then where to map the other six kinds of generator is forced if we are to satisfy the equations in Fig. 2. We sketch why this is so and refer the reader to [Pitts 2023, `fsRenset.agda`,`FullTransformationSemigroup.agda`] for the details.

[4] In fact $\tau_{a,i}$ and $\tau_{i,a}$ are equal, due to the symmetric nature of swapping.

First note that given $a, b \in \mathbb{A}\!\!\setminus$ and $x \in X$, choosing any $i \in \mathbb{N}$ with $i > x$, we have

$$\begin{aligned}
\varepsilon_{b,a} \odot x &= \varepsilon_{b,a} \odot (\{i \to b\}x) && \text{since } i > x \\
&= (\varepsilon_{b,a} \circ \varepsilon_{b,i}) \odot x && \text{by (50)} \\
&= (\varepsilon_{b,i} \circ \varepsilon_{i,a}) \odot x && \text{by TS}_7 \\
&= \{i \to b\}\{i \leftarrow a\}x && \text{by (50)} \\
&\triangleq [b \leftarrow\!\!\shortmid a]x && \text{from (21), since } i > x
\end{aligned}$$

There is a similar calculation for $\varepsilon_{j,i} \odot x$ versus $[i \mapsto j]x$ from (22). So we are forced to define

$$\varepsilon_{b,a} \odot x \triangleq [b \leftarrow\!\!\shortmid a]x \qquad \varepsilon_{j,i} \odot x \triangleq [i \mapsto j]x \tag{52}$$

One can check that (50) and (52) do indeed satisfy axioms TS_5–TS_8. In other words, X has a unique renset structure not just with respect to $\mathbb{A}\!\!\setminus$ (as in Proposition 2.17), but with respect to the whole of $\mathbb{N}\mathbb{A}\!\!\setminus$. Moreover, the fact that elements of X are finitely atom- and indexed-supported implies that each $x \in X$ is finitely supported with respect to the $\mathbb{N}\mathbb{A}\!\!\setminus$-renset structure of X. For if $A \in \text{Fin}\,\mathbb{A}\!\!\setminus$ atom-supports x and $n \in \mathbb{N}$ index-supports it, then the finite subset $n \cup A \triangleq \{i \in \mathbb{N} \mid i < n\} \cup A$ of $\mathbb{N}\mathbb{A}\!\!\setminus$ has the property that for every $a, b \in \mathbb{N}\mathbb{A}\!\!\setminus$ with $a \notin n \cup A$, $\varepsilon_{b,a} \odot x = x$. (See [Pitts 2023, `fsRenset.agda`].)

We saw in (29) how to define a name-swapping operation (for names from $\mathbb{A}\!\!\setminus$) satisfying the properties in Lemma 2.21, which correspond to TS_1–TS_4 and TS_9–TS_{13}. We can do the same thing with respect to names from the whole of $\mathbb{N}\mathbb{A}\!\!\setminus$: given $a, b \in \mathbb{N}\mathbb{A}\!\!\setminus$ and $x \in X$, we can well-define $\tau_{b,a} \odot x$ to be $\varepsilon_{b,c} \odot \varepsilon_{a,b} \odot \varepsilon_{c,a} \odot x$, where c is any element of $\mathbb{N}\mathbb{A}\!\!\setminus$ not in the support of x and not equal to a or b; and this definition does satisfy the axioms in Fig. 2. Furthermore $\tau_{b,a} \odot x$ is uniquely determined by those axioms: for if we choose c to also not be in the support of $\tau_{b,a} \odot x$, we have $\varepsilon_{b,c} \odot \tau_{b,a} \odot x = \tau_{b,a} \odot x$ and hence

$$\begin{aligned}
\tau_{b,a} \odot x &= \varepsilon_{b,b} \odot \varepsilon_{b,c} \odot \tau_{b,a} \odot x && \text{by TS}_5 \\
&= \varepsilon_{b,c} \odot \varepsilon_{c,b} \odot \tau_{b,a} \odot x && \text{by TS}_7 \\
&= \varepsilon_{b,c} \odot \varepsilon_{a,b} \odot \varepsilon_{c,a} \odot x && \text{by TS}_9
\end{aligned}$$

For the last sentence of the corollary, note that (51) holds when $m = \text{id}$; and if it holds for m_1 and m_2, it also holds for $m_1 \circ m_2$. Thus it suffices to check (51) as m ranges over the generators; and from above, we only need check that it holds for generators of the form $\varepsilon_{a,i}$ and $\varepsilon_{i,a}$. But (50) and the fact that f is a morphism of locally nameless sets together implies this. □

Lemma 3.9. *Suppose that X is a locally nameless set and that $\odot$ is as in Proposition 3.8. If $x \in X$ is atom-supported by $A \in \text{Fin}\,\mathbb{A}\!\!\setminus$ and index-supported by $n \in \mathbb{N}$, writing $n \cup A$ for $\{0, \ldots, n-1\} \cup A \subseteq \mathbb{N}\mathbb{A}\!\!\setminus$, we have for all $m, m' \in (\mathrm{T}_{\mathbb{N}\mathbb{A}\!\!\setminus})_{\text{fin}}$*

$$(\forall a \in n \cup A,\ m(a) = m'(a)) \implies m \odot x = m' \odot x \tag{53}$$

Proof. We proceed by induction on the size of the finite set $[m{\neq}m']$ from (48); it is finite because it is contained in $[m{\neq}\text{id}] \cup [m'{\neq}\text{id}]$ and we are assuming m and m' are finite functions. If this set is empty, then $m = m'$ and (53) holds trivially. So suppose $\forall a \in n \cup A,\ m(a) = m'(a)$, that $a_0 \in [m{\neq}m']$ (and hence $a_0 \notin n \cup A$) and that the result holds for all $m'', m''' \in (\mathrm{T}_{\mathbb{N}\mathbb{A}\!\!\setminus})_{\text{fin}}$ with $[m''{\neq}m''']$ of size one less than $[m{\neq}m']$. We have to show that $m \odot x = m' \odot x$.

Pick some $a_1 \in \mathbb{N}\mathbb{A}\!\!\setminus$ not in the finite subset $n \cup A \cup \{a_0\} \cup [m{\neq}\text{id}] \cup [m'{\neq}\text{id}]$ and consider $m'' \triangleq m \circ \varepsilon_{a_1,a_0}$ and $m''' \triangleq m' \circ \varepsilon_{a_1,a_0}$ in $(\mathrm{T}_{\mathbb{N}\mathbb{A}\!\!\setminus})_{\text{fin}}$. Thus m'' maps a_0 to a_1 and otherwise acts like m (since $a_1 \notin [m{\neq}\text{id}]$); and similarly for m'''. Therefore $[m''{\neq}m'''] = [m{\neq}m'] - \{a_0\}$ has strictly smaller size than $[m{\neq}m']$; and furthermore $\forall a \in n \cup A,\ m''(a) = m(a) = m'(a) = m'''(a)$. Hence

by induction hypothesis we have $m'' \odot x = m''' \odot x$. Since x is index-supported by n and atom-supported by A, $n \cup A$ is a support for x regarded as an $\mathbb{INA}$-renset (as in the proof of Proposition 3.8); so since $a_0 \notin n \cup A$ we have $\varepsilon_{a_1,a_0} \odot x = x$. Therefore $m'' \odot x \triangleq (m \circ \varepsilon_{a_1,a_0}) \odot x = m \odot \varepsilon_{a_1,a_0} \odot x = m \odot x$; and similarly $m''' \odot x = m' \odot x$. So $m \odot x = m'' \odot x = m''' \odot x = m' \odot x$, as required for the induction step. □

We can now construct an inverse for the functor $F : (\mathbf{Set}^{\mathrm{T}_{\mathbb{INA}}})_{\mathrm{fs}} \to \mathbf{Lns}$. Give a locally nameless set X, define a function $\_ \cdot \_ : \mathrm{T}_{\mathbb{INA}} \times X \to X$ as follows: for each $m \in \mathrm{T}_{\mathbb{INA}}$ and $x \in X$ define

$$m \cdot x \triangleq m' \odot x \tag{54}$$

where $\odot$ is as in Proposition 3.8 and $m' \in (\mathrm{T}_{\mathbb{INA}})_{\mathrm{fin}}$ is any finite endofunction satisfying $\forall a \in n \cup A,\ m(a) = m'(a)$ with $n \in \mathbb{N}$ an index-support for x and $A \in \mathrm{Fin}\,\mathbb{A}$ an atom-support for it. This definition makes sense because:

- If n and A are some index- and atom-supports for x, then we can take m' to be the finite endofunction
$$m'(a) \triangleq \begin{cases} m(a) & \text{if } a \in n \cup A \\ a & \text{if } a \notin n \cup A \end{cases} \tag{55}$$
- (54) is independent of the choice of n, A, m'. For suppose x also has index- and atom-supports n' and A' and that $m'' \in (\mathrm{T}_{\mathbb{INA}})_{\mathrm{fin}}$ satisfies $\forall a \in n' \cup A',\ m(a) = m''(a)$. It is easy to see from the definitions in Sect. 2.3 and 2.4 that x is also index-supported by $n'' \triangleq \min\{n, n'\}$ and atom-supported by $A'' \triangleq A \cap A'$; furthermore by assumption on m' and m'' we have $\forall a \in n'' \cup A'',\ m'(a) = m(a) = m''(a)$. Therefore $m' \odot x = m'' \odot x$ holds by Lemma 3.9.

So (54) does define a function $\_ \cdot \_ : \mathrm{T}_{\mathbb{INA}} \times X \to X$. We show that it is an associative and unitary action of the monoid $\mathrm{T}_{\mathbb{INA}}$ on X.

If $m_1, m_2 \in \mathrm{T}_{\mathbb{INA}}$ and $x \in X$, letting n and A be index- and atom-supports for x, then by definition $m_1 \cdot x = m_1' \odot x$ for some $m_1' \in (\mathrm{T}_{\mathbb{INA}})_{\mathrm{fin}}$ that agrees with m_1 on $n \cup A$. We can choose $n' \geq n$ and $A' \supseteq A$ so that n' index-supports and A' atom-supports $m_1' \odot x$ and furthermore $n' \cup A \supseteq \{m_1'(a) \mid a \in n \cup A\}$. Then by definition $m_2 \cdot (m_1' \odot x) = m_2' \odot (m_1' \odot x)$ for some $m_2' \in (\mathrm{T}_{\mathbb{INA}})_{\mathrm{fin}}$ that agrees with m_2 on $n' \cup A'$. Thus $m_2 \circ m_1$ agrees with $m_2' \circ m_1'$ on $n \cup A$ and hence by definition $(m_2 \circ m_1) \cdot x = (m_2' \circ m_1') \odot x$. So using the fact that $\odot$ is associative we have $(m_2 \circ m_1) \cdot x = (m_2' \circ m_1') \odot x = m_2' \odot (m_1' \odot x) = m_2 \cdot (m_1' \odot x) = m_2 \cdot (m_1 \cdot x)$.

For the unitary property, if $x \in X$, then since id is finite and agrees with itself on any support of x we have by definition and the unitary property of $\odot$ that $\mathrm{id} \cdot x = \mathrm{id} \odot x = x$.

Thus from $X \in \mathbf{Lns}$ we have constructed $\boxed{G(X)} \triangleq (X, \cdot)$ in $\mathbf{Set}^{\mathrm{T}_{\mathbb{INA}}}$. $G(X)$ is actually in $(\mathbf{Set}^{\mathrm{T}_{\mathbb{INA}}})_{\mathrm{fs}}$: for all $x \in X$, if $n \in \mathbb{N}$ index-supports x and $A \in \mathrm{Fin}\,\mathbb{A}$ atom-supports it, then we claim that $n \cup A$ finitely supports x in $(X, \cdot)$ in the sense of Definition 3.4. For if $m_1, m_2 \in \mathrm{T}_{\mathbb{INA}}$ satisfy $\forall a \in n \cup A,\ m_1(a) = m_2(a)$, then by definition $m_1 \cdot x = m_1' \odot x$ and $m_2 \cdot x = m_2' \odot x$ for some $m_1', m_2' \in (\mathrm{T}_{\mathbb{INA}})_{\mathrm{fin}}$ that agree with m_1 and m_2 respectively on $n \cup A$. So m_1' and m_2' are also equal when restricted to $n \cup A$ and hence by Lemma 3.9 we have $m_1' \odot x = m_2' \odot x$; and therefore $m_1 \cdot x = m_2 \cdot x$.

Note that if $f : X \to Y$ in $\mathbf{Lns}$, then for all $x \in X$ and $m \in \mathrm{T}_{\mathbb{INA}}$ we have $f(m \cdot x) = f(m' \odot x) = m' \odot (f\,x) = m \cdot (f\,x)$ by (54), (51) and the fact that n (respectively A) index-supports (respectively atom-supports) $f\,x$ when it does so for x. Therefore G extends to a functor $\mathbf{Lns} \to (\mathbf{Set}^{\mathrm{T}_{\mathbb{INA}}})_{\mathrm{fs}}$ whose action on morphisms is the identity. We will show that it is a two-sided inverse for the functor F.

If X is a locally nameless set, then for each $x \in X$, $i \in \mathbb{N}$ and $a \in \mathbb{A}$, since $\varepsilon_{a,i}$ and $\varepsilon_{i,a}$ are in $(\mathrm{T}_{\mathbb{INA}})_{\mathrm{fin}}$ from (50) and (54) we have $\varepsilon_{a,i} \cdot x = \varepsilon_{a,i} \odot x = \{i \to a\}x$ and $\varepsilon_{i,a} \cdot x = \varepsilon_{i,a} \odot x = \{i \leftarrow a\}x$. Therefore $F(G(X)) = X$ in $\mathbf{Lns}$.

Conversely, if $(X, \cdot) \in (\mathbf{Set}^{\mathrm{T}_{\mathbb{N}\mathbb{A}\setminus}})_{\mathrm{fs}}$, let us write $\_ \cdot_{G(F(X))} \_$ for the $\mathrm{T}_{\mathbb{N}\mathbb{A}\setminus}$-action on X that gives $G(F(X))$. We will show that it is equal to $\_ \cdot \_$ and hence that $G(F(X)) = X$ in $(\mathbf{Set}^{\mathrm{T}_{\mathbb{N}\mathbb{A}\setminus}})_{\mathrm{fs}}$.

To see this, given $m \in \mathrm{T}_{\mathbb{N}\mathbb{A}\setminus}$ and $x \in X$, suppose x is finitely supported by $S \in \mathrm{Fin}\,\mathbb{N}\mathbb{A}\setminus$. Choose some $n \in \mathbb{N}$ and $A \in \mathrm{Fin}\,\mathbb{A}\setminus$ so that $n \cup A \supseteq S$ and let $m' \in (\mathrm{T}_{\mathbb{N}\mathbb{A}\setminus})_{\mathrm{fin}}$ be defined as in (55). Note that n index-supports and A atom-supports x in $F(X)$. For if $i \geq n$ then $i \notin S$ so that for any $a \in \mathbb{A}\setminus$ we have $\forall c \in S,\ \varepsilon_{a,i}(c) = c = \mathrm{id}(c)$ and hence $\varepsilon_{a,i} \cdot x = \mathrm{id} \cdot x$ in $(X, \cdot)$; thus in $F(X)$ we have $\{i \to a\}x \triangleq \varepsilon_{a,i} \cdot x = \mathrm{id} \cdot x = x$. Similarly, if $a \in \mathbb{A}\setminus - A$ then $a \notin S$ and we get $\{0 \leftarrow a\}x = x$ in $F(X)$. Since m and m' agree on $n \cup A$ it follows from definition (54) that $m \cdot_{G(F(X))} x = m' \odot x$. By the uniqueness part of Proposition 3.8, the action $\_ \odot \_$ constructed from the locally nameless set $F(X)$ is equal to $\_ \cdot \_$. Therefore $m \cdot_{G(F(X))} x = m' \odot x = m' \cdot x$; but since m and m' agree on S which supports x in $(X, \cdot)$ we also have $m' \cdot x = m \cdot x$. Thus altogether we have $m \cdot_{G(F(X))} x = m \cdot x$ for all $m \in \mathrm{T}_{\mathbb{N}\mathbb{A}\setminus}$ and $x \in X$ and hence $G(F(X)) = X$.

We have now proved that the functors $F : (\mathbf{Set}^{\mathrm{T}_{\mathbb{N}\mathbb{A}\setminus}})_{\mathrm{fs}} \to \mathbf{Lns}$ and $G : \mathbf{Lns} \to (\mathbf{Set}^{\mathrm{T}_{\mathbb{N}\mathbb{A}\setminus}})_{\mathrm{fs}}$ are mutually inverse on objects. Since they both act on morphisms as the identity, they are in fact mutually inverse as functors. Thus we have completed the proof that $\mathbf{Lns}$ and $(\mathbf{Set}^{\mathrm{T}_{\mathbb{N}\mathbb{A}\setminus}})_{\mathrm{fs}}$ are isomorphic categories. □

Remark 3.10. Note that the single-renaming functions $\varepsilon_{a,i}$ and $\varepsilon_{i,a}$ (for $a \in \mathbb{A}\setminus$ and $i \in \mathbb{N}$) mentioned in the proof of Theorem 3.5 do not generate the whole monoid $\mathrm{T}_{\mathbb{N}\mathbb{A}\setminus}$. For one thing, compositions of them never take us out of the submonoid consisting of endofunctions $m : \mathbb{N}\mathbb{A}\setminus \to \mathbb{N}\mathbb{A}\setminus$ for which $[m \neq \mathrm{id}]$ is finite. But they do not even generate that submonoid, because any composition of a non-empty list of such endofunctions is always non-injective. Something more is needed to generate all the finite endofunctions, for example the single-transposition (swapping) functions (47); see Proposition 3.7. Nevertheless, the theorem shows that the single-renaming functions $\varepsilon_{a,i}$ and $\varepsilon_{i,a}$ together with the axioms in Fig. 1 completely characterise the action of all endofunctions (finite or not) of $\mathbb{N} \cup \mathbb{A}\setminus$ on finitely supported objects.

3.4 Shift Functor [Pitts 2023, `Shift.agda`]

In the locally nameless representation of syntax, opening and closing operations have their indices shifted up by one when they pass underneath (unary) binding constructs. Correspondingly, given an arbitrary locally nameless set X we can form a new one $\boxed{\uparrow X}$ that represents unary binding of names independently of any particular syntax. The underlying set of $\uparrow X$ is just that of X, but its opening/closing operations are given by

$$\{i \to^{+} a\}x \triangleq \{i+1 \to a\}x \qquad \{i \leftarrow^{+} a\}x \triangleq \{i+1 \leftarrow a\}x \tag{56}$$

Since $\_ + 1$ preserves $\neq$, these operation satisfy the axioms in Fig. 1 and hence $\uparrow X$ is an oc-set. It inherits the finite support properties (Definition 2.9) from X and so is also a locally nameless set. For given $x \in X$, if $A \in \mathrm{Fin}\,\mathbb{A}\setminus$ atom-supports x in X , it also atom-supports x in $\uparrow X$ (by virtue of Lemma 2.4); and if $i \succ x$ holds in X, then $i \dot{-} 1 \succ x$ holds in $\uparrow X$ (where $\dot{-}$ indicates truncated subtraction).

Note that the mapping $X \mapsto \uparrow X$ extends to a functor $\boxed{\uparrow : \mathbf{Lns} \to \mathbf{Lns}}$. The action of $\uparrow$ on a morphism $f : X \to Y$ is the identity at the level of sets and functions: $\uparrow f$ is just the function f, which does indeed satisfy (40) with respect to $\{\_ \to^{+} \_\}$ and $\{\_ \leftarrow^{+} \_\}$, because of the way they are defined in terms of $\{\_ \to \_\}$ and $\{\_ \leftarrow \_\}$.

We noted in Sect. 3.2 that the forgetful functor $\mathbf{Lns} \to \mathbf{Set}$ creates certain limits and colimits. From the way in which the functor $\uparrow : \mathbf{Lns} \to \mathbf{Lns}$ is defined, we have the following result, which will be useful in Sect. 4 when considering initial algebras of endofunctors on $\mathbf{Lns}$.

Proposition 3.11. $\uparrow : \mathbf{Lns} \to \mathbf{Lns}$ *preserves finite limits and filtered colimits.* □

3.5 Weakening Abstractions

An advantage of the nominal approach to dealing with syntax with binders [Pitts 2013], compared with the use of de Bruijn indices, is that a name in some context of use does not change when the context is weakened, whereas an index may have to. Since notations for explicit weakening and calculations with them can be very burdensome (either to the implementor of a proof tool, or to a user rolling their own metatheory within a general-purpose interactive theorem-prover) an approach that enables weakening to be invisible is attractive.[5] We will show that locally nameless sets take this one step further, by also making implicit the operations for abstracting extra fresh names.

For each nominal set X there is a nominal set of name-abstractions $[\mathbb{A}]X$ whose elements are equivalence classes of pairs $(a, x) \in \mathbb{A}\times X$ for a syntax-independent form of α-equivalence; see [Pitts 2013, Chapter 4] where the equivalence class of (a, x) is written $\langle a\rangle x$. Iterating this construction, for each $n \in \mathbb{N}$ one gets nominal sets $[\mathbb{A}]^n X$ of n-ary abstractions $\langle a_1, \ldots, a_n\rangle x$ (where $a_1, \ldots, a_n$ is an n-tuple of mutually distinct atoms).

Given $m \leq n$, each element of $[\mathbb{A}]^m X$ can be coerced into an element of $[\mathbb{A}]^n X$ by padding out the m-ary abstraction with $n - m$ fresh names. However, these coercions are not identity functions. By contrast the analogous construction for locally nameless sets is just an inclusion: if X is locally nameless, since $m \leq n$ we have $\mathrm{lc}_m(X) \subseteq \mathrm{lc}_n(X)$ (by Lemma 2.6). We sketch how to make precise the relationship between $[\mathbb{A}]^n X$ and $\mathrm{lc}_n(X)$.

We noted in Remark 2.20 that every locally nameless set supports a notion of name-swapping that makes it into a nominal set. Morphisms of locally nameless sets preserve the swapping operation and hence one gets a faithful functor $I : \mathbf{Lns} \to \mathbf{Nom}$ into the category **Nom** of nominal sets and equivariant functions. If $X \in \mathbf{Nom}$ and $Y \in \mathbf{Lns}$, from a morphism $f : X \to I(Y)$ in **Nom** we get a function $f^{(1)} : [\mathbb{A}]X \to Y$ well defined by $f^{(1)}(\langle a\rangle x) = {}^{\backslash a}(f\, x)$, using the locally nameless abstraction operation from (13). Iterating this we have functions $f^{(n)} : [\mathbb{A}]^n X \to Y$. One can show that $n \mapsto [\mathbb{A}]^n X$ is a diagram over $(\mathbb{N}, \leq)$ whose (filtered) colimit $F(X) \triangleq \mathrm{colim}_n [\mathbb{A}]^n X$ has opening and closing operations making it into a locally nameless set; and the functions $f^{(n)}$ induce a function $\hat{f} : F(X) \to Y$ which is a morphism in **Lns**. In this way we get a left adjoint $F : \mathbf{Nom} \to \mathbf{Lns}$ that glues together the n-ary name abstractions of elements of $X \in \mathbf{Nom}$ to make a locally nameless set. It is not hard to see from its definition that F preserves finite limits; therefore the adjunction $F \dashv I$ constitutes a geometric morphism $\mathbf{Lns} \to \mathbf{Nom}$. (In fact F has a left adjoint, given by taking the locally closed part of a locally nameless set, so the geometric morphism is *essential* [Johnstone 1977, Defn. 1.16]; and it is a *surjection* [*loc. cit.*, Defn. 4.11] because I is faithful.)

4 INITIAL ALGEBRA SEMANTICS WITH LOCALLY NAMELESS SETS

In this section we show that the usual inductive definition of syntax using the locally nameless representation of binders is a special case of the notion of initial algebra for endofunctors on **Lns** involving the shift functor.

4.1 Binding Signatures [Pitts 2023, `BindingSignature.agda`]

For simplicity we make use of the notion of *binding signature* of Plotkin [1990], rather than the slightly more flexible[6] notion of *nominal signature* from [Pitts 2013; Urban et al. 2004]. Recall from [Fiore et al. 1999] that a binding signature $\Sigma = (Op, ar)$ is given by a set of *operations* Op and an *arity* function $ar : Op \to \mathrm{List}\,\mathbb{N}$, the idea being that an operation $\mathrm{c} \in Op$ of arity $ar(\mathrm{c}) = [n_1, \ldots, n_k]$

[5] Nominal methods are not the only candidate for avoiding "weakening hell"; a synthetic approach in the sense of Sect. 4.3 could also achieve that.

[6] because it allows explicit name-sorts and multi-argument name-binding scopes

takes k arguments and binds n_i names in its i^{th} argument (for $i = 1, \ldots, k$). For example, the binding signature for untyped λ-terms has operations, lam and app, with arities $ar(\mathsf{lam}) = [1]$ and $ar(\mathsf{app}) = [0, 0]$.

Each binding signature $\Sigma = (Op, ar)$ determines an endofunctor $\boxed{\Sigma^{\uparrow} : \mathbf{Lns} \to \mathbf{Lns}}$ using the shift functor from Sect. 3.4 to interpret name-binding:

$$\Sigma^{\uparrow} X \triangleq \textstyle\sum_{\mathsf{c} \in Op} \uparrow^{ar(\mathsf{c})} X \tag{57}$$

In this definition, for each $X \in \mathbf{Lns}$ and $\vec{n} = [n_1, \ldots, n_k] \in \mathrm{List}\,\mathbb{N}$, the locally nameless set $\uparrow^{\vec{n}} X$ is given by the finite product

$$\uparrow^{\vec{n}} X \triangleq \uparrow^{n_1} X \times \cdots \times \uparrow^{n_k} X \tag{58}$$

where for each $n \in \mathbb{N}$, the locally nameless set $\uparrow^n X$ is the n^{th} iteration of the functor $\uparrow$ at X:

$$\uparrow^0 X \triangleq X \qquad\qquad \uparrow^{n+1} X \triangleq \uparrow(\uparrow^n X) \tag{59}$$

For example if Σ is the binding signature for untyped λ-term mentioned above, then $\Sigma^{\uparrow} : \mathbf{Lns} \to \mathbf{Lns}$ is the functor whose action on objects sends each $X \in \mathbf{Lns}$ to $\uparrow X + (X \times X)$.

4.2 Free $\Sigma^{\uparrow}$-Algebras

Recall that for each binding signature Σ there is a set of equivalence classes of terms over Σ modulo α-equivalence of bound names (using the concrete syntax for terms from [Fiore et al. 1999, Sect. 2], for example). This quotient set is in bijection with an inductively defined set using the locally nameless representation [Charguéraud 2012] (one first defines all locally nameless terms and then cuts down to the inductively defined subset of locally closed ones). Here we show that one can obtain the same thing by forming the free $\Sigma^{\uparrow}$-algebra in **Lns** on the object $\mathbb{N}\mathbb{A}\backslash$ (Example 2.10) and then taking its equationally defined locally closed part (Definition 2.14). To see this, we first recall some standard results about free algebras of endofunctors.

Given a functor $F : \mathbf{C} \to \mathbf{C}$ on a category $\mathbf{C}$ and an object $V \in \mathbf{C}$, the *free F-algebra on V* is given by an object $F[V] \in \mathbf{C}$ equipped with morphisms $V \xrightarrow{\iota} F[V] \xleftarrow{\alpha} F(F[V])$ with the universal property that for any other such diagram $V \xrightarrow{f} X \xleftarrow{g} F X$ in $\mathbf{C}$, there is a unique morphism $h : F[V] \to X$ making the following diagram commute

$$\begin{array}{ccccc} V & \xrightarrow{\iota} & F[V] & \xleftarrow{\alpha} & F(F[V]) \\ {\scriptstyle \mathrm{id}}\downarrow & & \downarrow{\scriptstyle h} & & \downarrow{\scriptstyle Fh} \\ V & \xrightarrow[f]{} & X & \xleftarrow[g]{} & F X \end{array} \tag{60}$$

If $\mathbf{C}$ has finite coproducts and colimits of chains of some limit ordinal length λ that are preserved by F, then by the classical theorem of Adámek [1974], the free F-algebra $F[V]$ exists and is given by iterating the functor $V + F(\_)$ starting at the initial object to get chain of length λ and taking its colimit. Here we know that **Lns** has filtered colimits and $\Sigma^{\uparrow}$ preserves them (because of Proposition 3.11). Therefore the free $\Sigma^{\uparrow}$-algebra $\Sigma^{\uparrow}[\mathbb{N}\mathbb{A}\backslash] \in \mathbf{Lns}$ can be constructed as the colimit of the countable chain

$$\emptyset \to \mathbb{N}\mathbb{A}\backslash + \Sigma^{\uparrow}\emptyset \to \mathbb{N}\mathbb{A}\backslash + \Sigma^{\uparrow}(\mathbb{N}\mathbb{A}\backslash + \Sigma^{\uparrow}\emptyset) \to \mathbb{N}\mathbb{A}\backslash + \Sigma^{\uparrow}(\mathbb{N}\mathbb{A}\backslash + \Sigma^{\uparrow}(\mathbb{N}\mathbb{A}\backslash + \Sigma^{\uparrow}\emptyset)) \to \cdots \tag{61}$$

There is a more useful description of $\Sigma^{\uparrow}[\mathbb{N}\mathbb{A}\backslash]$. Consider the functor $\Sigma^{\mathrm{Id}} : \mathbf{Set} \to \mathbf{Set}$ defined like (57)–(59) but using the identity functor $\mathrm{Id} : \mathbf{Set} \to \mathbf{Set}$ rather than the abstraction functor $\uparrow : \mathbf{Lns} \to \mathbf{Lns}$. Thus

$$\Sigma^{\mathrm{Id}} X = \textstyle\sum_{\mathsf{c} \in Op} X^{|ar(\mathsf{c})|} \quad \text{(where } |\vec{n}| \in \mathbb{N} \text{ indicates the length of a list } \vec{n} \in \mathrm{List}\,\mathbb{N}) \tag{62}$$

is the finitary polynomial functor derived from the signature Σ by ignoring the binding information of arities. Although the free algebra $\Sigma^{\mathrm{Id}}[\mathbb{N} \cup \mathbb{A}]$ can be constructed as a countable colimit in **Set**

$$\emptyset \longrightarrow \mathbb{N} \cup \mathbb{A} + \Sigma^{\mathrm{Id}}\,\emptyset \longrightarrow \mathbb{N} \cup \mathbb{A} + \Sigma^{\mathrm{Id}}(\mathbb{N} \cup \mathbb{A} + \Sigma^{\mathrm{Id}}\,\emptyset) \longrightarrow \cdots \tag{63}$$

a possibly more familiar description of this set is as an inductively defined set A with constructors

$$\mathrm{var} : \mathbb{N} \cup \mathbb{A} \to A \qquad\qquad \mathrm{c} : A \times \cdots \times A \to A \qquad (\mathrm{c} \in \mathit{Op}) \tag{64}$$

where the number of arguments of the constructor $\mathrm{c} \in \mathit{Op}$ is the length of the list $\mathit{ar}(\mathrm{c})$. For example, writing bvar and fvar for the restriction of var to $\mathbb{N}$ and $\mathbb{A}$ respectively, in the case that Σ is the signature for λ terms, we get the inductively defined set Λ in (9). Writing $U : \mathbf{Lns} \to \mathbf{Set}$ for the forgetful functor, note that the diagrams

$$\begin{array}{ccc} \mathbf{Lns} & \xrightarrow{\uparrow} & \mathbf{Lns} \\ {\scriptstyle U}\downarrow & & \downarrow{\scriptstyle U} \\ \mathbf{Set} & \xrightarrow[\mathrm{Id}]{} & \mathbf{Set} \end{array} \qquad \begin{array}{ccc} \mathbf{Lns} \times \mathbf{Lns} & \xrightarrow{\times} & \mathbf{Lns} \\ {\scriptstyle U\times U}\downarrow & & \downarrow{\scriptstyle U} \\ \mathbf{Set} \times \mathbf{Set} & \xrightarrow[\times]{} & \mathbf{Set} \end{array} \qquad \begin{array}{ccc} \mathbf{Lns} \times \mathbf{Lns} & \xrightarrow{+} & \mathbf{Lns} \\ {\scriptstyle U\times U}\downarrow & & \downarrow{\scriptstyle U} \\ \mathbf{Set} \times \mathbf{Set} & \xrightarrow[+]{} & \mathbf{Set} \end{array}$$

commute and U maps $\mathbb{N}\mathbb{A}\backslash$ to $\mathbb{N} \cup \mathbb{A}\backslash$. It follows that U sends the colimit of (61) in **Lns** to the colimit of (63) in **Set**. In other words, the underlying set of the free $\Sigma^{\uparrow}$-algebra on $\mathbb{N}\mathbb{A}\backslash$, $\Sigma^{\uparrow}[\mathbb{N}\mathbb{A}\backslash] \in \mathbf{Lns}$, is just the set A inductively defined in (64). Furthermore, since the constructor functions in (64) are the images under U of morphisms of locally named sets, the open/close operations necessarily satisfy the properties that are used by Charguéraud [2012, sections 3.1 and 3.2] to inductively define these operations:

$$\begin{gathered} \{i \to a\}(\mathrm{bvar}\ j) = \mathit{if}\ i = j\ \mathit{then}\ \mathrm{fvar}\ a\ \mathit{else}\ \mathrm{bvar}\ j \qquad\qquad \{i \leftarrow a\}(\mathrm{bvar}\ j) = \mathrm{bvar}\ j \\ \{i \to a\}(\mathrm{fvar}\ b) = \mathrm{fvar}\ b \qquad \{i \leftarrow a\}(\mathrm{fvar}\ b) = \mathit{if}\ a = b\ \mathit{then}\ \mathrm{bvar}\ i\ \mathit{else}\ \mathrm{fvar}\ b \\ \{i \to a\}\mathrm{c}(t_1, \ldots, t_k) = \mathrm{c}(\{i + n_1 \to a\}t_1, \ldots \{i + n_k \to a\}t_k) \\ \{i \leftarrow a\}\mathrm{c}(t_1, \ldots, t_k) = \mathrm{c}(\{i + n_1 \leftarrow a\}t_1, \ldots, \{i + n_k \leftarrow a\}t_k) \quad (\text{where } \mathit{ar}(\mathrm{c}) = [n_1, \ldots, n_k]) \end{gathered}$$

So we have:

Theorem 4.1. *For every binding signature Σ, the usual locally nameless representation of the syntax of Σ-terms modulo α-equivalence is given by the locally nameless set $\Sigma^{\uparrow}[\mathbb{N}\mathbb{A}\backslash]$ that is the free $\Sigma^{\uparrow}$-algebra on the locally nameless set $\mathbb{N}\mathbb{A}\backslash$ of indices or atoms.* □

Not only does $\Sigma^{\uparrow}[\mathbb{N}\mathbb{A}\backslash]$ agree with the set of terms mod α over a binding signature Σ defined in locally nameless style, but also the usual inductive notions of "free variable" and "local closure" agree with the equational notions that can be defined in any locally nameless set using the definitions from Sects 2.3 and 2.5, as the next two propositions show. Their proofs proceed by induction on the structure of t in the inductively defined set A (64); see the Agda development [Pitts 2023, `BindingSignature.agda`] for details.

Proposition 4.2 (Freshness vs free variables). *Given a binding signature Σ, for all elements t of the underlying set A (64) of the free $\Sigma^{\uparrow}$-algebra on $\mathbb{N}\mathbb{A}\backslash$, define the finite set $\mathrm{fv}(t) \in \mathrm{Fin}\,\mathbb{A}\backslash$ of* free variables *as usual for the locally nameless representation of syntax:*

$$\mathrm{fv}(\mathrm{bvar}\ i) = \emptyset \qquad \mathrm{fv}(\mathrm{fvar}\ a) = \{a\} \qquad \mathrm{fv}(\mathrm{c}(t_1, \ldots, t_k)) = \mathrm{fv}(t_1) \cup \cdots \cup \mathrm{fv}(t_k) \tag{65}$$

Then for all $a \in \mathbb{A}\backslash$ and $t \in A$ we have: $a \# t \iff a \notin \mathrm{fv}(t)$. In particular, the validity of the finite atom-support property $\forall a,\ a \# t$ is witnessed by $\mathrm{fv}(t) \in \mathrm{Fin}\,\mathbb{A}\backslash$. □

Proposition 4.3 (Index-support vs local closedness). *Given a binding signature Σ, for all elements t of the underlying set A (64) of the free $\Sigma^{\uparrow}$-algebra on $\mathbb{N}\!\mathbb{A}$, the predicate*

$$\mathrm{lc\_at}\, i\, t \qquad (\text{“}t \text{ is closed at level } i\text{”})$$

from [Charguéraud 2012, Sect. 3.3] is inductively defined by the rules:

$$\frac{j < i}{\mathrm{lc\_at}\, i\, (\mathrm{bvar}\, j)} \qquad \frac{i \in \mathbb{N} \qquad a \in \mathbb{A}}{\mathrm{lc\_at}\, i\, (\mathrm{fvar}\, a)} \qquad \frac{\mathrm{lc\_at}(i + n_1)\, t_1 \qquad \cdots \qquad \mathrm{lc\_at}(i + n_k)\, t_k}{\mathrm{lc\_at}\, i\, (\mathrm{c}(t_1, \ldots, t_k))}$$

Then for all $i \in \mathbb{N}$ and $t \in A$ we have: $i > t \iff \mathrm{lc\_at}\, i\, t$. In particular, the finite index-support property $\exists i \in \mathbb{N}.\ i > t$ is witnessed by the fact that $\mathrm{lv}\, t > t$, where $\mathrm{lv}\, t \in \mathbb{N}$ is given by:

$$\mathrm{lv}(\mathrm{bvar}\, i) = i + 1 \qquad \mathrm{lv}(\mathrm{fvar}\, a) = 0 \qquad \mathrm{lv}(\mathrm{c}(t_1, \ldots, t_k)) = \max\{\mathrm{lv}(t_1), \ldots, \mathrm{lv}(t_k)\}$$

□

Example 4.4 (Denotations via initiality). Let **Dom** denote the category of Scott domains and Scott continuous functions [Scott 1982] and suppose that $D \in \mathbf{Dom}$ is a domain with a retraction onto its own exponential D^D, so that there are Scott continuous functions $\mathrm{lm} : D^D \to D$ and $\mathrm{ap} : D \to D^D$ with $\mathrm{ap} \circ \mathrm{lm} = \mathrm{id}$. These can be used to give denotational semantics to the untyped λ-calculus in the following way. Let $D^{\mathbb{N}\cup\mathbb{A}}$ denote the product in **Dom** of $(\mathbb{N} \cup \mathbb{A})$-many copies of D and define

$$\mathrm{C}(D) \triangleq \mathbf{Dom}(D^{\mathbb{N}\cup\mathbb{A}}, D) \tag{66}$$

to be the set of Scott continuous functions $D^{\mathbb{N}\cup\mathbb{A}} \to D$. We think of the elements $\rho \in D^{\mathbb{N}\cup\mathbb{A}}$ as environments mapping indices and atoms to their denotations in D; and the elements $c \in \mathrm{C}(D)$ as (semantic) continuations mapping such environments to elements of D. Every function $m : \mathbb{N}\cup\mathbb{A} \to \mathbb{N} \cup \mathbb{A}$ induces a Scott continuous re-indexing function $m^* : D^{\mathbb{N}\cup\mathbb{A}} \to D^{\mathbb{N}\cup\mathbb{A}}$ in **Dom** and hence by pre-composition, a function $m^{**} : \mathrm{C}(D) \to \mathrm{C}(D)$ in **Set**. The mapping $m \mapsto m^{**}$ is functorial and hence $\mathrm{C}(D)$ inherits the structure of an oc-set from that of $\mathbb{N}\!\mathbb{A}$ (Sect. 2.2). For example, the opening operation produces from a continuation $c \in \mathrm{C}(D)$ a continuation $\{i \to a\}c \in \mathrm{C}(D)$ that maps each $\rho \in D^{\mathbb{N}\cup\mathbb{A}}$ to $c(\rho \circ \varepsilon_{a,i})$, where $\varepsilon_{a,i} : \mathbb{N} \cup \mathbb{A} \to \mathbb{N} \cup \mathbb{A}$ is as in (46); and similarly for closing operations.

Since $\mathrm{C}(D)$ is an oc-set we can form the subset $\mathrm{C}(D)_{\mathrm{fs}}$ of finitely atom- and index-supported elements of $\mathrm{C}(D)$ to obtain a locally nameless set. (Recall from Sect. 3.2 that $X \mapsto X_{\mathrm{fs}}$ gives a right adjoint to the inclusion **Lns** $\hookrightarrow$ oc-**Set**.) Although $\mathrm{C}(D)_{\mathrm{fs}}$ has nothing *per se* to do with syntax, it does have a binding operation, in the sense that there is a morphism in **Lns**

$$\overline{\mathrm{lm}} : \uparrow(\mathrm{C}(D)_{\mathrm{fs}}) \to \mathrm{C}(D)_{\mathrm{fs}} \qquad \overline{\mathrm{lm}}\, c\, \rho \triangleq \mathrm{lm}(\lambda d \in D.\ c[\rho, d]) \quad (c \in \uparrow(\mathrm{C}(D)_{\mathrm{fs}}), \rho \in D^{\mathbb{N}\cup\mathbb{A}}) \tag{67}$$

where $[\_, \_] : D^{\mathbb{N}\cup\mathbb{A}} \times D \cong D^{\mathbb{N}\cup\mathbb{A}\cup\{*\}} \cong D^{\mathbb{N}\cup\mathbb{A}}$ in **Dom** is induced by bijection $\mathbb{N}\cup\mathbb{A}\cup\{*\} \cong \mathbb{N}\cup\mathbb{A}$ (we assume $* \notin \mathbb{N}\!\mathbb{A}$) that that shifts indices up by 1, leaves atoms unchanged and maps the unique element $*$ of $\{*\}$ to index 0. (Note that it follows that $\lambda d \in D.\ c[\rho, d]$ is Scott continuous, hence is an element of D^D to which we can apply $\mathrm{lm} : D^D \to D$ in the definition of $\overline{\mathrm{lm}}$.) To see that (67) is a morphism in **Lns** one just has to check that it commutes with the opening and closing operations (since the fact that it yields finitely supported elements follows from that); but this is a routine calculation from the definitions. Similarly, but more simply, there are morphisms in **Lns**

$$\overline{\mathrm{ap}} : \mathrm{C}(D)_{\mathrm{fs}} \times \mathrm{C}(D)_{\mathrm{fs}} \to \mathrm{C}(D)_{\mathrm{fs}} \qquad \overline{\mathrm{ap}}(c, c')\, \rho \triangleq \mathrm{ap}(c\, \rho)(c'\rho) \quad (c, c' \in \mathrm{C}(D)_{\mathrm{fs}}, d \in D^{\mathbb{N}\cup\mathbb{A}}) \tag{68}$$

$$\overline{\mathrm{vr}} : \mathbb{N}\!\mathbb{A} \to \mathrm{C}(D)_{\mathrm{fs}} \qquad \overline{\mathrm{vr}}\, u\, \rho \triangleq \rho(u) \quad (u \in \mathbb{N}\!\mathbb{A}, \rho \in D^{\mathbb{N}\cup\mathbb{A}}) \tag{69}$$

Hence $\mathrm{C}(D)_{\mathrm{fs}}$ is an $\Sigma^{\uparrow}$-algebra for the binding signature Σ for λ-terms mentioned at the end of Sect. 4.1; and by Theorem 4.1 there is a unique algebra morphism $[\![\_]\!] : \Sigma^{\uparrow}[\mathbb{N}\!\mathbb{A}] \to \mathrm{C}(D)_{\mathrm{fs}}$ in

Lns with $[\![\_]\!] \circ \mathsf{var} = \overline{\mathsf{vr}}$. We noted above that in general the underlying set of $\Sigma^{\uparrow}[\mathbb{N}\mathbb{A}\backslash]$ is just the inductively defined set of terms for the binding signature Σ defined in locally nameless style; and in this particular case it is the locally nameless set Λ from Example 2.11. Furthermore, when restricted to locally closed elements $t \in \Lambda$, for each environment ρ the element $[\![t]\!]\rho \in D$ recovers the usual denotational semantics of the term t in the domain D. This example of denotations-via-initiality for locally nameless syntax should be compared with Fiore et al. [1999, end of Sect. 2] for the mathematics of the de Bruijn representation, with Popescu [2022, Sect. 5.3] for his finitely supported renaming sets, and with Pitts [2006, Sect. 6.3] for the nominal approach (where one has to use some ingenuity (unfortunately) to get the requisite recursion principle to apply).

Example 4.5 (Capture-avoiding substitution). In the preceding example, the fact that $\mathrm{C}(D)_{\mathrm{fs}}$ is a locally nameless set does not depend much upon the specific nature of the category **Dom**—the same definition (66) will work for an object D in any locally small category where the $(\mathbb{N} \cup \mathbb{A}\backslash)$-fold product of D exists. **Lns** is such a category: the $(\mathbb{N} \cup \mathbb{A}\backslash)$-fold product of $X \in \mathbf{Lns}$ is necessarily given by the locally nameless coreflection $(X^{\mathbb{N}\cup\mathbb{A}\backslash})_{\mathrm{fs}}$ of the product $X^{\mathbb{N}\cup\mathbb{A}\backslash}$ in oc-**Set**, which is itself created by the forgetful functor to **Set**, that is, is the set of functions from $\mathbb{N} \cup \mathbb{A}\backslash$ to X with opening/closing operations induced pontwise from those of X. Taking X to be the locally nameless set Λ of λ-terms (Example 2.11), we have a locally nameless set $\mathrm{C}(\Lambda)_{\mathrm{fs}}$, where now

$$\mathrm{C}(\Lambda) \triangleq \mathbf{Lns}((\Lambda^{\mathbb{N}\cup\mathbb{A}\backslash})_{\mathrm{fs}}, \Lambda) \tag{70}$$

is the set of morphisms $(\Lambda^{\mathbb{N}\cup\mathbb{A}\backslash})_{\mathrm{fs}} \to \Lambda$ in **Lns**, equipped with the opening/closing operations given by $\{i \to a\}c = c \circ \varepsilon^{*}_{a,i}$ and $\{i \leftarrow a\}c = c \circ \varepsilon^{*}_{i,a}$. We think of the elements $\sigma \in (\Lambda^{\mathbb{N}\cup\mathbb{A}\backslash})_{\mathrm{fs}}$ as (finite) *substitutions* of terms for indices and atoms. The locally nameless set $\mathrm{C}(\Lambda)_{\mathrm{fs}}$ carries the structure of a $\Sigma^{\uparrow}$-algebra (when Σ is the signature for λ-terms):

$$\overline{\mathsf{lam}} : \uparrow(\mathrm{C}(\Lambda)_{\mathrm{fs}}) \to \mathrm{C}(\Lambda)_{\mathrm{fs}} \qquad \overline{\mathsf{lam}}\, c\, \sigma \triangleq \mathsf{lam}(c\,\sigma) \tag{71}$$

$$\overline{\mathsf{app}} : \mathrm{C}(\Lambda)_{\mathrm{fs}} \times \mathrm{C}(\Lambda)_{\mathrm{fs}} \to \mathrm{C}(\Lambda)_{\mathrm{fs}} \qquad \overline{\mathsf{app}}\,(c, c')\,\sigma \triangleq \mathsf{app}(c\,\sigma, c'\sigma) \tag{72}$$

(where $c, c' \in \uparrow(\mathrm{C}(\Lambda)_{\mathrm{fs}})$ and $\sigma \in (\Lambda^{\mathbb{N}\cup\mathbb{A}\backslash})_{\mathrm{fs}}$) and there is a morphism

$$\overline{\mathsf{var}} : \mathbb{N}\mathbb{A}\backslash \to \mathrm{C}(\Lambda)_{\mathrm{fs}} \qquad \overline{\mathsf{var}}\, u\, \sigma \triangleq \sigma(u) \quad (u \in \mathbb{N}\mathbb{A}\backslash, \sigma \in (\Lambda^{\mathbb{N}\cup\mathbb{A}\backslash})_{\mathrm{fs}}) \tag{73}$$

Therefore by Theorem 4.1 there is a unique $\Sigma^{\uparrow}$-algebra morphism $sub : \Lambda \to \mathrm{C}(\Lambda)_{\mathrm{fs}}$ in **Lns** with $sub \circ \mathsf{var} = \overline{\mathsf{var}}$. This agrees with the usual, inductively defined notion of capture-avoiding substitution for the locally nameless representation (*cf.* [Charguéraud 2012, Sect. 3.5])

$$\begin{array}{rcl} sub\,(\mathsf{bvar}\, i)\,\sigma & = & \sigma\, i \\ sub\,(\mathsf{fvar}\, a)\,\sigma & = & \sigma\, a \\ sub\,(\mathsf{lam}\, t)\,\sigma & = & \mathsf{lam}(sub\, t\, \sigma) \\ sub\,(\mathsf{app}(t, t'))\,\sigma & = & \mathsf{app}(sub\, t\, \sigma, sub\, t'\, \sigma) \end{array} \tag{74}$$

(where $t, t' \in \Lambda$, $a \in \mathbb{A}\backslash$ and $i \in \mathbb{N}$). This ignores the usual yoga of shifting dangling de Bruijn indices within t when passing under binding constructs like lam—a selling point of the locally nameless approach. However, as a result $sub\, t\, \sigma$ is only a correct definition of *capture-avoiding* substitution if one restricts to locally closed t and σ. In practice it is possible to maintain such local closedness invariants while proving properties of locally nameless syntax, often through the use of cofinite quantification [Aydemir et al. 2008; Charguéraud 2012; McKinna and Pollack 1999]. However, doing so means that one has to look outside the category **Lns**, as we discuss next.

4.3 Towards Synthetic Locally Nameless

There are several results like Theorem 4.1 in the literature on the mathematics of syntax with binding constructs. For example, the three influential papers that appeared together in LICS 1999 [Fiore et al. 1999; Gabbay and Pitts 1999; Hofmann 1999] all feature, in various categories, initial algebra characterizations of syntax modulo α-equivalence using respectively nominal, de Bruijn and weak higher-order representations. In the first case the category is the Schanuel topos (equivalently, the topos of nominal sets) and in the second two cases it is toposes of presheaves for categories of contexts and variable-renamings. Work on various kinds of sets-equipped-with-a-renaming-action [Gabbay and Hofmann 2008; Popescu 2022] also feature such results. In all cases there is an emphasis, more or less explicitly, on deriving from the initiality property (60) more useable recursion and induction principles for reasoning about represented syntax. The utility of such recursion and induction principles partly depends upon how many constructs of interest can be expressed within the internal logic of the categories involved. For example, most constructs of interest are invariant under name permutation and many have finite support; and so nominal recursion/induction [Pitts 2013, Chapter 8] is widely applicable in principle[7]. Compared with name permutation, renaming preserves fewer constructs of interest (particularly not ones involving logical negation) and this could make an initial algebra result in categories based on renaming less easy to apply; but see [Popescu 2022] for the surprisingly good state of the art. The situation for **Lns** is worse: its internal logic does not support *by itself* some of the key constructs needed for the locally nameless approach. For example, taking the locally closed part (Definition 2.14) of a locally nameless set yields a nominal set, but not a locally nameless one (since it is not usually invariant under the operation of closing an atom with an index); and yet as mentioned in Examples 4.4 and 4.5, use of the initiality theorem only gives correct notions of denotation and substitution when restricting attention to locally closed elements.

Nevertheless, a *synthetic* account of locally nameless representations and computations with them is a desirable goal (for the general reasons set out by Sterling [2021, Sect. 0.6]). In other words one would like an expressive type theory or logic featuring axioms directly capturing the key features of the mathematical analysis that this paper has provided. To be useful, expressiveness has to be balanced by simplicity of the axiomatic notions and existence of a straightforward mathematical model of them. For the latter, **Lns** by itself will not work; but a topos obtained from it may work, for example by considering **Lns** relative to the topos of nominal sets as in Sect. 3.5. The internal logic could then be a two-level type theory of some kind (some types being nominal-set-like, some being locally-nameless-set-like) with the levels connected by a modality for local closure. To be practically useful such a type theory will have to feature "Barendregt-enhanced" [8] recursion and induction principles involving the cofinite quantifier (Definition 2.1), analogous to those considered by Pitts [2006] and Popescu [2022, Theorem 11]. We leave investigation of such a synthetic account for future work.

5 AGDA DEVELOPMENT

Agda [2023] was used to develop the theory of locally nameless sets and check some of the proofs. The Agda code [Pitts 2023] mainly targets proofs that involve equational reasoning combined with the use of atoms and indices that are sufficiently fresh (via cofinite quantification). Agda was chosen because of the author's familiarity with it; as [Charguéraud 2012] reports, most developments so far using the locally nameless representation have employed Coq or Isabelle/HOL. In this section

[7] But hampered in practice by its need to satisfy various kinds of freshness side-conditions.
[8] The name points to a relationship with the informal "variable condition" of Barendregt [1984, Appendix A].

we discuss some of the features of our Agda development, assuming some familiarity with Agda's concrete syntax.

5.1 Atoms and Cofinite Quantification [Pitts 2023, Unfinite.agda]

For the decidable, "unfinite" (1) set 𝔸 one could just take atoms to be in bijection with natural numbers. We prefer another representation that focuses on the crucial property that given a finite set of atoms A, there is an atom (call it `new A`) not equal to any of the atoms in A. We make this the definition of "atom" and use an inductive type `𝔸` with a single constructor `new` of type `Fset 𝔸 → 𝔸`, where the elements of the indexed inductive type `Fset X` are trees representing finite subsets of `X`. One can prove that the type `𝔸` has decidable equality, so that its equality `_≡_ : 𝔸 → 𝔸 → Set` has a Boolean complement `_≠_ : 𝔸 → 𝔸 → Set`. The definition of `𝔸` makes it equivalent to a type of well-founded trees (a W-type [Nordström et al. 1990, Chapter 15]); and using well-founded induction one can prove the unfiniteness property

```
unfinite : (A : Fset 𝔸) → new A ∉ A
```

where `_∉_` is the non-membership predicate, inductively defined using decidability of equality in `𝔸`. Its definition makes use of Agda's *instance arguments* (indicated by double braces `{{_}}`), a special kind of implicit argument searched for by an algorithm separate from the one used for normal implicit arguments. These instance arguments are the Agda equivalent of Haskell type class constraints. Here they allow a certain amount of automation for proving non-membership properties, in particular for the witnesses involved in proving cofinite quantifications (Definition 2.1), using the following definition:

```
record ∀𝔸 (P : 𝔸 → Set) : Set where
  constructor ∀𝔸i
  field
    ∀𝔸e₁ : Fset 𝔸
    ∀𝔸e₂ : (a : 𝔸){{_ : a ∉ ∀𝔸e₁}} → P a
open ∀𝔸 public
syntax ∀𝔸 (λ a → P) = ∀𝔸 a : 𝔸 , P
```

5.2 Locally Nameless Types [Pitts 2023, oc-Sets.agda,Support.agda]

The definition of oc-sets and locally nameless sets is straightforward, using a naive unbundled approach to type classes (via Agda's instance arguments, mentioned above). An oc-set structure for a type `X` is an element of a record type `oc (X : Set)` with fields

```
_~>_ : ℕ → 𝔸 → X → X
_<~_ : ℕ → 𝔸 → X → X
oc₁ : ∀ i a b x → (i ~> a)((i ~> b)x) ≡ (i ~> b)x
```

and also fields $\mathtt{oc}_2, \ldots \mathtt{oc}_9$ corresponding to the other axioms in Fig. 1. Defining the associated notions of freshness (3) and local closedness (5)

```
_#_ : {X : Set}{{_ : oc X}} → 𝔸 → X → Set
a # x = (0 <~ a)x ≡ x
_>_ : {X : Set}{{_ : oc X}} → ℕ → X → Set
i > x = (j : ℕ){{_ : j ≥ i}} → ∑ a : 𝔸 , ((j ~> a)x ≡ x)
```

then a locally nameless set structure for a type `X` is an element of the type `lns(X)`, where

```
record lns (X : Set) : Set where
  field
    {{ocSet}} : oc X
    asupp : (x : X) → ∀ a : 𝔸 , (a # x)
    isupp : (x : X) → ∑ i : ℕ , (i ≻ x)
```

5.3 Shift Functor [Pitts 2023, Shift.agda]

The shift functor (Sect. 3.4) changes oc-set structure (and locally nameless structure) without changing the underlying type

```
oc↑ : {X : Set}{{_ : oc X}} → oc X
_~>_ {{oc↑}} i a x = (i +1 ~> a)x
_<~_ {{oc↑}} i a x = (i +1 <~ a)x
```

(the rest of the definition is omitted). This makes the functor conveniently invisible when building inductive datatypes involving it (such as `Trm` Σ below). However, it does disrupt automatic inference of structure, since Agda takes exception if it finds two or more (or no) instances of a structure for a given type. An alternative would be to make $\uparrow X$ have an underlying set that is a record type isomorphic to X.

5.4 Binding Signatures [Pitts 2023, BindingSignature.agda]

Rather than using lists of numbers as arities as in Sect. 4.1, we found it more convenient to use functional arrays:

```
record Array (X : Set) : Set where
  field
    length : ℕ
    index : Fin length → X
```

where `Fin : ℕ → Set` is the usual parameterised inductive type of finite sets. We define the type of Plotkin-style binding signatures to be

```
record Sig : Set₁ where
  field
    Op : Set
    ar : Op → Array ℕ
```

The inductive type of terms over such a signature (64) is then

```
data Trm (Σ : Sig) : Set where
  var : ℕ𝔸 → Trm Σ
  op  : (c : Op Σ) → (Fin(length(ar Σ c)) → Trm Σ) → Trm Σ
```

where ℕ𝔸 is the disjoint union of ℕ and 𝔸. For each signature Σ, the type `Trm` Σ has the structure of a locally nameless set making it the free $\Sigma^{\uparrow}$-algebra on ℕ𝔸 (Theorem 4.1); and an element's support as determined by the oc-algebra structure agrees with the usual inductively defined notions (Propositions 4.2 and 4.3). The downside of using functional arrays rather than lists is that the proof of these facts uses function extensionality. However, that principle is needed anyway when proving the uniqueness part of Proposition 3.8 and the initiality property (60). We postulate function extensionality in our Agda development, but an alternative might be to work in a type theory in which it is provable, such as Cubical Agda [Vezzosi et al. 2019]. However, the current Agda

development does make use of Coquand's original form of dependent pattern matching [Coquand 1992] for which uniqueness of identity proofs (UIP) is provable; in other words the development uses the Agda option `--with-K`. It may be possible to eliminate use of UIP and to transfer the development to Cubical Agda, but we have not investigated this.

5.5 "Mere" Existence

The use of a record type (or equivalently, a Σ-type) to define cofinite quantification in Sect. 5.1 means that, given a proof of `∀ a : A , P`, applying the projection $\forall e_1$ to it we obtain an explicit witness for the finite subset of `𝔸` on which `P` may not hold. This corresponds to how cofinite quantification occurs in mechanized metatheory using the locally nameless representation. For example, a proof of typing in a system like that in [Charguéraud 2012, Fig. 1] contains explicit occurrences of finite sets witnessing the various cofinite quantifications in the proof (and hence in particular there can be different proofs of the same typing judgment). Similarly, when defining locally nameless structure `lns X` we used a Σ-type for the finite index-support property `isupp x` of `x : X`, instead of an existenial quantifier as in (8). These choices suffice for constructive proofs of the results presented in Sects 2, 3.1–3.4 and 4; but not for Sect. 3.5, unless one moves to a classical setting in which witnesses for infinite families of existentially quantified statements can be found (for example via the Axiom of Choice).

The reason for this has to do with properties of the notion of "finite support". The theory of nominal sets in [Pitts 2013] makes the assumption that every element of a nominal set possesses a *smallest* finite set of atoms that supports it. Swan [2017] has shown that this assumption implies the non-constructive *weak limited principle of omniscience* (WLPO), which says that given a property P of natural numbers, if we can decide for each $x \in \mathbb{N}$ whether or not $P\,x$ holds, then we can decide whether or not the set $\{x \in \mathbb{N} \mid P\,x\}$ is empty. There is strong (but unpublished) evidence that a rich constructive theory of nominal sets can be developed if one just requires that for every element of a constructive nominal set there merely exists some finite set of atoms that supports it (without assuming that there is a smallest such, or even assuming that there is a function mapping each element to a support set for it).

The situation for locally nameless sets is likely to be the same. Therefore, where the current Agda formalization uses the strong form of existential quantification given by Σ-types, one should try instead to work with "mere" existence [Univalent Foundations Program 2013, Sect. 3.6]. For example, the reflection of nominal sets into locally nameless sets sketched in Sect. 3.5 involves a countably infinite colimit, which will be constructed by taking a quotient by a suitable equivalence relation. Given an equivalence class, to see that there merely exists a finite support for it, it suffices to pick a representative of the class and use its finite support; but without some form of choice principle one would not have the *function* assigning a support to each equivalence class that would be required if one uses the definition of support in the current development.

6 CONCLUSION

The equational axiomatization of opening and closing in Fig. 1 is quite simple, although it does have its subtleties (axioms OC_8 and OC_9). It came as a big surprise to the author that it suffices to derive so many of the key concepts of the locally nameless representation of syntax with binders, independently of any particular syntax. The results in this paper show how to automatically derive from a signature specification a large part of the infrastructure described in [Charguéraud 2012, Sect. 3.9]. At the moment what remains of that infrastructure is the use of recursion and induction principles for ordinary inductively defined sets (rather than locally nameless sets) to define specific functions and relations needed for a particular development. It may be that this can be synthesised into a new "locally nameless logic", as discussed in Sect. 4.3.

Less speculatively, the algebraic treatment of opening and closing has allowed us to relate the mathematics of the locally nameless representation to nominal representations that are based on the use of renaming functions, by showing that categories that have been used in the literature for modelling those forms of representation are in fact equivalent to the category of locally nameless sets introduced here.

For the first time we have a syntax-free account of the locally nameless version of name binding, via the shift functor on locally nameless sets. Before this work one only knew what locally nameless *syntax* means. As Example 4.4 illustrates, now one can make sense of "locally nameless semantics". This could enable the pleasant properties of the locally nameless representation to be used for situations where syntax and semantics get mixed up – for example in proofs of normalization-by-evaluation [Berger and Schwichtenberg 1991] (see Pitts [2006, Sect. 6], for example). The equational treatment of the locally nameless representation may also help to improve automated support in theorem provers for this method of developing formal metatheory of programming languages and logics. For example, it may be possible to combine our Agda development with the work of Escot and Cockx [2022] to extend their encoding of datatypes to ones involving the shift functor, together with a generic construction of locally nameless set structure for Agda datatypes that have such an encoding.

ACKNOWLEDGMENTS

The author is grateful to the anonymous reviewers for their helpful comments and suggestions.

REFERENCES

Jiří Adámek. 1974. Free Algebras and Automata Realizations in the Language of Categories. *Commentationes Mathematicae Universitatis Carolinae* 15, 4 (1974), 589–602. http://eudml.org/doc/16649

Agda 2023. The Agda Wiki. https://wiki.portal.chalmers.se/agda/Main/HomePage.

Brian Aydemir, Arthur Charguéraud, Benjamin C. Pierce, Randy Pollack, and Stephanie Weirich. 2008. Engineering Formal Metatheory. In *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (San Francisco, California, USA) *(POPL '08)*. Association for Computing Machinery, New York, NY, USA, 3–15. https://doi.org/10.1145/1328438.1328443

Henk P. Barendregt. 1984. *The Lambda Calculus: Its Syntax and Semantics* (revised ed.). North-Holland.

Ulrich Berger and Helmut Schwichtenberg. 1991. An Inverse of the Evaluation Functional for Typed λ-Calculus. In *6th Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, 203–211. https://doi.org/10.1109/LICS.1991.151645

Arthur Charguéraud. 2012. The Locally Nameless Representation. *Journal of Automated Reasoning* 49, 3 (2012), 363–408. https://doi.org/10.1007/s10817-011-9225-2

Jesper Cockx. 2021. 1001 Representations of Syntax with Binding. (Nov. 2021). Blog post https://jesper.sikanda.be/posts/1001-syntax-representations.html.

Thierry Coquand. 1992. Pattern Matching with Dependent Types. In *Proceedings of the 1992 Workshop on Types for Proofs and Programs, Båstad, Sweden*, Bengt Nordström, Kent Petersson, and Gordon D. Plotkin (Eds.). 66–79.

Nicolaas G. de Bruijn. 1972. Lambda Calculus Notation with Nameless Dummies, a Tool for Automatic Formula Manipulation, with Application to the Church-Rosser Theorem. *Indagationes Mathematicae* 34 (1972), 381–392. https://doi.org/10.1016/1385-7258(72)90034-0

Lucas Escot and Jesper Cockx. 2022. Practical Generic Programming over a Universe of Native Datatypes. *Proc. ACM Program. Lang.* 6, ICFP, Article 113 (aug 2022), 25 pages. https://doi.org/10.1145/3547644

Marcelo P. Fiore, Gordon P. Plotkin, and Daniele Turi. 1999. Abstract Syntax and Variable Binding. In *14th Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, 193–202. https://doi.org/10.1109/LICS.1999.782615

Murdoch J. Gabbay and Martin Hofmann. 2008. Nominal Renaming Sets. In *Logic for Programming, Artificial Intelligence, and Reasoning, 15th International Conference, LPAR 2008, Doha, Qatar, November 22–27, 2008. Proceedings (Lecture Notes in Artificial Intelligence, Vol. 5330)*. Springer, 158–173. https://doi.org/10.1007/978-3-540-89439-1_11

Murdoch J. Gabbay and Andrew M. Pitts. 1999. A New Approach to Abstract Syntax Involving Binders. In *14th Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, 214–224. https://doi.org/10.1109/LICS.1999.782617

Murdoch J. Gabbay and Andrew M. Pitts. 2002. A New Approach to Abstract Syntax with Variable Binding. *Formal Aspects of Computing* 13 (2002), 341–363. https://doi.org/10.1007/s001650200016

Olexandr Ganyushkin and Volodymyr Mazorchuk. 2009. *Classical Finite Transformation Semigroups, An Introduction.* Algebra and Applications, Vol. 9. Springer, London.

Martin Hofmann. 1999. Semantical Analysis of Higher-Order Abstract Syntax. In *14th Annual Symposium on Logic in Computer Science.* IEEE Computer Society Press, 204–213. https://doi.org/10.1109/LICS.1999.782616

Nagayoshi Iwahori and Nabuko Iwahori. 1974. On a Set of Generating Relations of the Full Transformation Semigroups. *Journal of Combinatorial Theory (A)* 16 (1974), 147–158. https://doi.org/10.1016/0097-3165(74)90040-5

Peter T. Johnstone. 1977. *Topos Theory.* Number 10 in LMS Mathematical Monographs. Academic Press.

Peter T. Johnstone. 2002. *Sketches of an Elephant, A Topos Theory Compendium, Volumes 1 and 2.* Number 43–44 in Oxford Logic Guides. Oxford University Press.

James McKinna and Randy Pollack. 1999. Some Type Theory and Lambda Calculus Formalised. *Journal of Automated Reasoning* 23 (1999), 373–409. https://doi.org/10.1023/A:1006294005493

Bengt Nordström, Kent Petersson, and Jan M. Smith. 1990. *Programming in Martin-Löf's Type Theory.* Oxford University Press.

Frank Pfenning and Conal Elliott. 1988. Higher-Order Abstract Syntax. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation.* ACM Press, 199–208. https://doi.org/10.1145/960116.54010

A. M. Pitts. 2003. Nominal Logic, A First Order Theory of Names and Binding. *Information and Computation* 186 (2003), 165–193. https://doi.org/10.1016/S0890-5401(03)00138-X

Andrew M. Pitts. 2006. Alpha-Structural Recursion and Induction. *Journal of the ACM* 53 (2006), 459–506. https://doi.org/10.1145/1147954.1147961

Andrew M. Pitts. 2013. *Nominal Sets: Names and Symmetry in Computer Science.* Cambridge Tracts in Theoretical Computer Science, Vol. 57. Cambridge University Press.

Andrew M. Pitts. 2015. Nominal Presentation of Cubical Sets Models of Type Theory. In *20th International Conference on Types for Proofs and Programs (TYPES 2014) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 39).* Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 202–220. https://doi.org/10.4230/LIPIcs.TYPES.2014.202

Andrew M. Pitts. 2023. Agda code accompanying "Locally Nameless Sets". https://doi.org/10.5281/zenodo.7121016 Browsable code: https://amp12.github.io/LocallyNamelessSets/.

Gordon D. Plotkin. 1990. An Illative Theory of Relations. In *Situation Theory and its Applications, Volume 1*, R. Cooper, Mukai, and J. Perry (Eds.). CSLI Lecture Notes, Vol. 22. Stanford University, 133–146.

Andrei Popescu. 2022. Rensets and Renaming-Based Recursion for Syntax with Bindings. In *Automated Reasoning : 11th International Joint Conference, IJCAR 2022, Haifa, Israel, August 8-10, 2022, Proceedings (Lecture Notes in Computer Science, Vol. 13385).* Springer International Publishing AG, 618–639. https://doi.org/10.1007/978-3-031-10769-6_36

Dana S. Scott. 1982. Domains for Denotational Semantics. In *Automata, Languages and Programming, Proceedings 1982 (Lecture Notes in Computer Science, Vol. 140)*, Mogens Nielson and Erik M. Schmidt (Eds.). Springer Berlin Heidelberg, 577–610. https://doi.org/10.1007/BFb0012801

Sam Staton. 2007. *Name-Passing Process Calculi: Operational Models and Structural Operational Semantics.* Ph. D. Dissertation. University of Cambridge. Available as University of Cambridge Computer Laboratory Technical Report Number UCAM-CL-TR-688.

Jonathan Sterling. 2021. *First Steps in Synthetic Tait Computability, The Objective Metatheory of Cubical Type Theory.* Ph. D. Dissertation. Carnegie Mellon University, School of Computer Science. CMU-CS-21-142.

Andrew Swan. 2017. Some Brouwerian Counterexamples Regarding Nominal Sets in Constructive Set Theory. *ArXiv e-prints* arXiv:1702.01556 [math.LO] (Feb. 2017), 9pp. https://doi.org/10.48550/arXiv.1702.01556

The Univalent Foundations Program. 2013. *Homotopy Type Theory: Univalent Foundations for Mathematics.* http://homotopytypetheory.org/book, Institute for Advanced Study.

Christian Urban, Andrew M. Pitts, and Murdoch J. Gabbay. 2004. Nominal Unification. *Theoretical Computer Science* 323 (2004), 473–497. https://doi.org/10.1016/j.tcs.2004.06.016

Christian Urban and Christine Tasson. 2005. Nominal Techniques in Isabelle/HOL. In *20th International Conference on Automated Deduction, CADE-20, Tallinn, Estonia, July 2005 (Lecture Notes in Computer Science, Vol. 3632).* Springer-Verlag, 38–53. https://doi.org/10.1007/11532231_4

Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. 2019. Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types. *Proc. ACM Program. Lang.* 3, ICFP, Article 87 (July 2019), 29 pages. https://doi.org/10.1145/3341691

Received 2022-07-07; accepted 2022-11-07