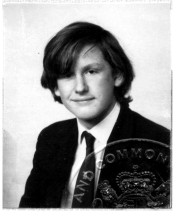
Locally Nameless Sets

Andrew Pitts
UNIVERSITY OF CAMBRIDGE

50th POPL, Boston 2023

the speaker while not
attending the 1st POPL



Plan

Context: development of the meta-theory of programming languages within interactive theorem provers (Agda, Coq, Isabelle/HOL, Lean, ...)

- Review of the locally nameless method for representing and computing with the syntax of languages involving binding constructs.
- A new mathematical foundation for it, independent of any particular language.
- Consequences of that new foundation.

Named, Nameless and Locally Nameless

illustrated by a running example from simply typed λ-calculus with products

named (nominal) terms: bound vars named, free vars named

$$\lambda y : B.\ (x, y)$$

Named, Nameless and Locally Nameless

illustrated by a running example from simply typed λ-calculus with products

named (nominal) terms: bound vars named, free vars named

$$x : A, y : A \quad \vdash \quad \lambda y : B.\, (x, y) : B \to A \times B$$
$$\lambda z : B.\, (x, z)$$

Named, Nameless and Locally Nameless

illustrated by a running example from simply typed λ-calculus with products

named (nominal) terms: bound vars named, free vars named

$$x : A, y : A \quad \vdash \quad [\lambda y : B.\, (x, y)]_\alpha : B \to A \times B$$ not purely inductive

α equiv class

Named, Nameless and Locally Nameless

illustrated by a running example from simply typed λ-calculus with products

named (nominal) terms: bound vars named, free vars named

$$x : A, y : A \quad \vdash \quad [\lambda y : B.\, (x, y)]_\alpha : B \to A \times B$$ not purely inductive

$$x : A, y : A, z : C \quad \vdash \quad [\lambda y : B.\, (x, y)]_\alpha : B \to A \times B$$ weakening invariant

Named, Nameless and Locally Nameless

illustrated by a running example from simply typed λ-calculus with products

named (nominal) terms: bound vars named, free vars named

nameless (de Bruijn) terms: bound vars indexed, free vars indexed (dangling)

$\lambda B.\ (2, 0)$ purely inductive

Named, Nameless and Locally Nameless

illustrated by a running example from simply typed λ-calculus with products

named (nominal) terms: bound vars named, free vars named

nameless (de Bruijn) terms: bound vars indexed, free vars indexed (dangling)

$$A, A \quad \vdash \quad \lambda B.\,(2, 0) : B \to A \times B \qquad \text{purely inductive}$$

$$A, A, C \quad \vdash \quad \lambda B.\,(2 + 1, 0) : B \to A \times B \qquad \text{not weakening invariant}$$

Named, Nameless and Locally Nameless

illustrated by a running example from simply typed λ-calculus with products

named (nominal) terms: bound vars named, free vars named

nameless (de Bruijn) terms: bound vars indexed, free vars indexed (dangling)

locally nameless terms: bound vars indexed, free vars named

$\lambda B.\ (x, 0)$ purely inductive

Named, Nameless and Locally Nameless

illustrated by a running example from simply typed λ-calculus with products

named (nominal) terms: bound vars named, free vars named

nameless (de Bruijn) terms: bound vars indexed, free vars indexed (dangling)

locally nameless terms: bound vars indexed, free vars named

$$x : A, y : A \quad \vdash \quad \lambda B.\,(x, 0) : B \to A \times B$$ purely inductive

$$x : A, y : A, z : C \quad \vdash \quad \lambda B.\,(x, 0) : B \to A \times B$$ weakening invariant

See Aydemir, Charguéraud, Pierce, Pollack & Weirich, POPL 2008
Charguéraud, *J. Automated Reasoning* 49(2012)363–408

Locally Nameless infrastructure

operations

opening	$\boxed{\{i \to x\} t}$	=	term obtained from t by replacing index i with variable x
closing	$\boxed{\{i \leftarrow x\} t}$	=	term obtained from t by replacing variable x with index i

relations

freshness	$\boxed{x \# t}$	=	variable x does not occur in term t
locally closed up to a level	$\boxed{i \succ t}$	=	all indices in term t are $< i$
locally closed	$\boxed{\mathrm{lc}\ t}$	=	$0 \succ t$ (i.e. t contains no indices)

Locally Nameless infrastructure

By restricting attention to locally closed terms, one can avoid error-prone index-shifting operations. E.g. for capture-avoiding substitution of u for x in t $\boxed{[x \to u]t}$, so long as $\mathrm{lc}\ u$ holds, it is correct to define it at lambda-abstractions by

$$[x \to u](\lambda\, A\, t) \triangleq \lambda\, A\, ([x \to u]t)$$

freshness $\boxed{x \# t}$ = variable x does not occur in term t

locally closed up to a level $\boxed{i \succ t}$ = all indices in term t are $< i$

locally closed $\boxed{\mathrm{lc}\ t}$ = $0 \succ t$ (i.e. t contains no indices)

Locally nameless infrastructure

The need to restrict to locally closed terms means that a locally closed binding form such as $\lambda\, A\, t$ should not be deconstructed to t (not locally closed), but rather to $\{0{\to}x\}t$ with x fresh.

E.g.

$$\frac{\text{И}x,\ (\Gamma, x : A \vdash \{0{\to}x\}t : B)}{\Gamma \vdash \lambda\, A\, t : A \to B}$$

cofinite quantifier

$\boxed{\text{И}x,\ \varphi(x)}$ = property $\varphi(x)$ holds for all but finitely many variables x

(cf. McKinna-Pollack | Gabbay-AMP nominal freshness quantifier)
Aydemir, Charguéraud, Pierce, Pollack & Weirich, POPL 2008
Charguéraud, *J. Automated Reasoning* 49(2012)363–408

A new syntax-independent theory

Up to now, locally nameless developments depend on the structure of the object language (case-by-case)

- $\{i \to x\} t$ and $\{i \leftarrow x\} t$ defined by recursion over term t
- $x \# t$ and $i \succ t$ defined by induction over term t

A new syntax-independent theory

Up to now, locally nameless developments depend on the structure of the object language (case-by-case)

New (surprising?) insight

[inspired by "nominal techniques", *ACM SIGLOG News*, 3(2016)57-72]

- opening/closing operations can be given an equational axiomatization
- freshness and local closedness can be defined in terms of opening/closing and И, with expected properties with respect to finitely supported objects

Locally nameless sets

Fix atomic *names* $x, y, z, \ldots \in \mathbb{A}$ ($\cong \mathbb{N}$) and disjoint *indices* $i, j, k, \ldots \in \mathbb{N}$.

Definition. A set S is locally nameless
if it comes equipped with two
('opening' and 'closing') functions

$$\{\_\to\_\}\_ \quad \{\_\leftarrow\_\}\_ : \mathbb{N} \to \mathbb{A} \to S \to S$$

satisfying nine equations
and the finite support property

$$\forall s \in S,\ (\text{И}x,\ x \# s) \wedge (\exists i,\ i \succ s)$$

freshness
$\forall i,\ \{i\leftarrow x\}s = s$

local closedness
$\forall j \geq i,\ \text{И}x,\ \{j\to x\}s = s$

Locally nameless sets

Fix atomic *names* $x, y, z, \ldots \in \mathbb{A}$ ($\cong \mathbb{N}$) and disjoint *indices* $i, j, k, \ldots \in \mathbb{N}$.

Definition. A set S is locally nameless
if it comes equipped with two
('opening' and 'closing') functions

$$\{\_\to\_\}\_ \quad \{\_\leftarrow\_\}\_ : \mathbb{N} \to \mathbb{A} \to S \to S$$

satisfying nine (5 mod duality) equations
and the finite support property

$$\forall s \in S,\ (\text{И}x,\ x \# s) \wedge (\exists i,\ i \succ s)$$

$$\{i\to x\}\{i\to y\}s = \{i\to y\}s$$
$$\{i\leftarrow x\}\{j\leftarrow x\}s = \{j\leftarrow x\}s$$
$$\{i\leftarrow x\}\{i\to x\}s = \{i\leftarrow x\}s$$
$$\{i\to x\}\{i\leftarrow x\}s = \{i\to x\}s$$
$$\{i\to x\}\{j\to y\}s = \{j\to y\}\{i\to x\}s \quad (i \neq j)$$
$$\{i\leftarrow x\}\{j\leftarrow y\}s = \{j\leftarrow y\}\{i\leftarrow x\}s \quad (x \neq y)$$
$$\{i\to x\}\{j\leftarrow y\}s = \{j\leftarrow y\}\{i\to x\}s \quad (i \neq j,\ x \neq y)$$
$$\{i\to y\}\{i\leftarrow x\}\{j\to y\}s = \{j\to y\}\{j\leftarrow x\}\{i\to x\}s$$
$$\{j\leftarrow x\}\{i\to x\}\{j\leftarrow y\}s = \{j\leftarrow y\}\{i\to y\}\{i\leftarrow x\}s$$

The opening/closing axioms imply that every locally nameless set has well-defined operations of

renaming

$\{i{\leftarrow}x\}a$ is given

$\{i{\rightarrow}x\}a$ is given

$$\{y{\leftarrow}x\}a \triangleq \{i{\rightarrow}y\}\{i{\leftarrow}x\}a \quad \text{for some/any } i \succ a$$

$$\{i{\rightarrow}j\}a \triangleq \cdots$$

swapping

$$(x\ y)a \triangleq \{j{\rightarrow}x\}\{i{\rightarrow}y\}\{j{\leftarrow}y\}\{i{\leftarrow}x\}a \quad \text{for some/any } i, j \succ a \text{ with } i \neq j$$

$$(i\ j)a \triangleq \cdots$$

$$(i\ x)a = (x\ i)a \triangleq \cdots$$

The opening/closing axioms imply that every locally nameless set has well-defined operations of

renaming

$$\{i{\leftarrow}x\}a \quad \text{is given}$$

$$\{i{\rightarrow}x\}a \quad \text{is given}$$

$$\{y{\leftarrow}x\}a \triangleq \{i{\rightarrow}y\}\{i{\leftarrow}x\}a \quad \text{for some/any } i \succ a$$

$$\{i{\rightarrow}j\}a \triangleq \cdots$$

swapping

$$(x\ y)a \triangleq \{j{\rightarrow}x\}\{i{\rightarrow}y\}\{j{\leftarrow}y\}\{i{\leftarrow}x\}a \quad \text{for some/any } i, j \succ a \text{ with } i \neq j$$

$$(i\ j)a \triangleq \cdots$$

$$(i\ x)a = (x\ i)a \triangleq \cdots$$

Theorem. Every locally nameless set is a renset [Popescu, IJCAR 2022] and (hence) a nominal set [AMP, CUP 2013].

Hence get the action of *any* function $\mathbb{N} \cup \mathbb{A} \to \mathbb{N} \cup \mathbb{A}$ (because of finite support property).

The category **Lns**

Theorem.
Isomorphism of categories:

$$\mathbf{Lns} \cong (\mathbf{Set}^{\mathrm{T}_\omega})_{\mathrm{fs}}$$

(Right-hand side is a known topos.)

We get a category $\boxed{\mathbf{Lns}}$ of locally nameless sets by taking morphisms to be functions that commute with the opening/closing functions.

The category **Lns**

Theorem.
Isomorphism of categories:

$$\mathbf{Lns} \cong (\mathbf{Set}^{\mathrm{T}_\omega})_{\mathrm{fs}}$$

(Right-hand side is a known topos.)

T_ω = monoid of all functions $\omega \to \omega$

We get a category $\boxed{\mathbf{Lns}}$ of locally nameless sets by taking morphisms to be functions that commute with the opening/closing functions.

The category **Lns**

Theorem.
Isomorphism of categories:

$$\mathbf{Lns} \cong (\mathbf{Set}^{\mathrm{T}_\omega})_{\mathrm{fs}}$$

(Right-hand side is a known topos.)

We get a category $\boxed{\mathbf{Lns}}$ of locally nameless sets by taking morphisms to be functions that commute with the opening/closing functions.

$\mathbf{Set}^{\mathrm{T}_\omega}$ = category of sets equipped with an action of T_ω

full subcategory whose objects are those T_ω-sets whose elements are finitely supported

The category **Lns**

Theorem.
Isomorphism of categories:

$$\mathbf{Lns} \cong (\mathbf{Set}^{\mathrm{T}_\omega})_{\mathrm{fs}}$$

(Right-hand side is a known topos.)

Proof involves several ingredients, one of which is a known presentation of *finite* full transformation monoids T_n [Iwahori & Iwahori, *J. Comb. Theory (A)*, 16(1974)147–158].

The category $\mathbf{Lns}$

Theorem.
Isomorphism of categories:

$$\mathbf{Lns} \cong (\mathbf{Set}^{\mathrm{T}_\omega})_{\mathrm{fs}}$$

(Right-hand side is a known topos.)

Corollary of the theorem's proof is that various categories that have been used to model renaming [Staton 2007; Gabbay & Hofmann 2008; Popescu 2022] are all equivalent to each other and to the category $\mathbf{Lns}$ of locally nameless sets.

Locally nameless sets

Fix atomic *names* $x, y, z, \ldots \in \mathbb{A}$ ($\cong \mathbb{N}$) and disjoint *indices* $i, j, k, \ldots \in \mathbb{N}$.

Definition. A set S is locally nameless if it comes equipped with two ('opening' and 'closing') functions

$$\{\_\to\_\}\_ \quad \{\_\leftarrow\_\}\_ : \mathbb{N} \to \mathbb{A} \to S \to S$$

satisfying nine equations and the finite support property

$$\forall s \in S,\ (\reflectbox{N} x,\ x \# s) \wedge (\exists i,\ i \succ s)$$

The abstract versions of $\{\to\}$, $\{\leftarrow\}$, $\#$ and $\succ$ agree with the usual concrete recursive/inductive definitions when A is an inductively defined set giving the locally nameless representation of some language.

So what did we gain?

Consequences

Binding has a generic, language-independent definition as a "shift" functor $\boxed{\uparrow : \mathbf{Lns} \to \mathbf{Lns}}$ given by shifting indices.

$$\begin{tikzcd} \uparrow S \arrow[r, bend right] \arrow[r, phantom, "\cong"] & S \arrow[l, bend right, "\mathrm{abs}"'] \end{tikzcd}$$

$$\begin{aligned} \{i{\to}x\}(\mathrm{abs}\, s) &\triangleq \mathrm{abs}(\{i{+}1{\to}x\}s) \\ \{i{\leftarrow}x\}(\mathrm{abs}\, s) &\triangleq \mathrm{abs}(\{i{+}1{\leftarrow}x\}s) \end{aligned}$$

Consequences

Binding has a generic, language-independent definition as a "shift" functor $\boxed{\uparrow : \mathbf{Lns} \to \mathbf{Lns}}$ given by shifting indices.

- Initial algebras for functors combining $\uparrow$ with $+$ and $\times$ recover locally nameless finitary syntax (see paper for details).

  E.g. initial algebra for functor $S \mapsto (\mathbb{N} \cup \mathbb{A}) + (S \times S) + \uparrow S$ is isomorphic to usual locally nameless datatype for untyped λ-terms.

Consequences

Binding has a generic, language-independent definition as a "shift" functor $\boxed{\uparrow : \mathbf{Lns} \to \mathbf{Lns}}$ given by shifting indices.

- Initial algebras for functors combining $\uparrow$ with $+$ and $\times$ recover locally nameless finitary syntax (see paper for details).
- There are lots of non-syntactic locally nameless sets.

  E.g. $S \triangleq ((\mathbb{N} \cup \mathbb{A} \to R) \to R)_{\mathrm{fs}}$, which has a binding operation $\uparrow S \to S$ if there is a function $(R \to R) \to R$.

  Potential for "locally nameless *semantics*" (applications to NBE?)

Consequences

Binding has a generic, language-independent definition as a "shift" functor $\boxed{\uparrow : \mathbf{Lns} \to \mathbf{Lns}}$ given by shifting indices.

- Initial algebras for functors combining $\uparrow$ with $+$ and $\times$ recover locally nameless finitary syntax (see paper for details).
- Potential for "locally nameless *semantics*" (applications to NBE?)
- [to do] Simple automation of locally nameless "boilerplate" within interactive theorem provers

  (Cf. LNgen (`ott`→Coq) and Autosubst (Coq library for nameless).)

Consequences

Binding has a generic, language-independent definition as a "shift" functor $\boxed{\uparrow : \mathbf{Lns} \to \mathbf{Lns}}$ given by shifting indices.

- Initial algebras for functors combining $\uparrow$ with $+$ and $\times$ recover locally nameless finitary syntax (see paper for details).
- Potential for "locally nameless *semantics*" (applications to NBE?)
- [to do] Simple automation of locally nameless "boilerplate" within interactive theorem provers

```
data S : Set where
  var : (ℕ + 𝔸) → S
  app : S × S → S
  lam : ↑ S → S
deriving LocallyNameless
```

Summary

- New insight: locally nameless can be founded on an algebra of opening/closing operations, independent of any object-level language.
- Cofinite quantification plays a key role.
- Mathematical status of the axioms.
- Data generic functor for binding.
  Before: locally nameless syntax.
  Now: locally nameless syntax and semantics.
- Potential for typeclass-style automation of locally nameless boiler plate.

Summary

- New insight: locally nameless can be founded on an algebra of opening/closing operations, independent of any object-level language.
- Cofinite quantification plays a key role.
- Mathematical status of the axioms.
- Data generic functor for binding.
  Before: locally nameless syntax.
  Now: locally nameless syntax and semantics.
- Potential for typeclass-style automation of locally nameless boiler plate.

End