TYPES+PCC 2014

Nominal Sets and Dependent Type Theory

Andrew Pitts

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Plan

$$\left\{ \begin{array}{c} \text{Type Theory} \\ \text{presheaf categories} \end{array} \right\} \text{audience listens to talk} \left\{ \begin{array}{c} \\ \\ \\ \end{array} \right\}$$

Plan

$\left\{ \begin{array}{c} \text{Type Theory} \\ \text{presheaf categories} \end{array} \right\}$ audience listens to talk $\left\{ \begin{array}{c} \text{nominal sets:} \\ \text{freshness} \\ \text{name abstraction} \end{array} \right\}$

aim to explain the notions of

freshness and **name-abstraction**

from the theory of nominal sets

and discuss two (on-going) applications involving dependent types:

1. Cubical sets model of Homotopy Type Theory.
2. A version of Type Theory with names, freshness and name-abstraction.

Freshness

What is a fresh name?

Possible definition: name a is fresh if it is not ‘stale’:

a is not equal to any name in the current (finite) set of used names (and we extend that set with a)

What is a fresh name?

Possible definition: name a is fresh if it is not 'stale':

a is not equal to any name in the current (finite) set of used names (and we extend that set with a)

- need to be able to test names for equality – that is the only attribute we assume names have (atomic names)

What is a fresh name?

Possible definition: name a is fresh if it is not 'stale':

a is not equal to any name in the current (finite) set of used names (and we extend that set with a)

- need to be able to test names for equality – that is the only attribute we assume names have (atomic names)
- freshness has a modal character – suggests using Kripke-Beth-Joyal (possible worlds) semantics with. . .

Presheaf semantics

$\mathbb{I}$ = category of finite ordinals
$n = \{0, 1, \ldots, n-1\}$
and injective functions

$$\mathbf{U} \in [\mathbb{I}, \mathbf{Set}]$$

Presheaf semantics

$\mathbb{I}$ = category of finite ordinals
$n = \{0, 1, \ldots, n-1\}$
and injective functions

$$\mathbf{U} \in [\mathbb{I}, \mathbf{Set}]$$

$[\mathbb{I}, \mathbf{Set}]$ = (covariant) presheaf category:
set-valued functors X & natural transformations.
$X\,n$ = set of objects (of some type)
possibly involving n distinct names

Presheaf semantics

$\mathbb{I}$ = category of finite ordinals
$n = \{0, 1, \ldots, n-1\}$
and injective functions

generic decidable object
U = inclusion functor:
$\mathsf{U}\, n = \{0, 1, \ldots, n-1\}$

$\mathsf{U} \in [\mathbb{I}, \mathbf{Set}]$

$[\mathbb{I}, \mathbf{Set}]$ = (covariant) presheaf category:
set-valued functors X & natural transformations.
$X\, n$ = set of objects (of some type)
possibly involving n distinct names

Generic decidable object

$\mathbf{U}$ is a 'decidable' object of the topos $[\mathbb{I}, \mathbf{Set}]$

> diagonal subobject $\mathbf{U} \rightarrowtail \mathbf{U} \times \mathbf{U}$ has a boolean complement $\neq \rightarrowtail \mathbf{U} \times \mathbf{U}$

Generic decidable object

U is a 'decidable' object of the topos $[\mathbb{I}, \mathbf{Set}]$

$$a =_{\mathsf{U}} b \;\wedge\; \neq(a,b) \;\Rightarrow\; \mathbf{false}$$
$$\mathbf{true} \;\Rightarrow\; a =_{\mathsf{U}} b \;\vee\; \neq(a,b)$$

Generic <u>infinite</u> decidable object

U is a 'decidable' object of the topos $[\mathbb{I}, \mathbf{Set}]$

$$a =_{\mathsf{U}} b \;\wedge\; \neq(a,b) \;\Rightarrow\; \mathbf{false}$$
$$\mathbf{true} \;\Rightarrow\; a =_{\mathsf{U}} b \;\vee\; \neq(a,b)$$

but it does not satisfy 'finite inexhaustibility'

$$\bigwedge_{0 \le i < j \le n} \neq(a_i, a_j) \;\Rightarrow\; \bigvee_{b:\mathsf{U}} \bigwedge_{0 \le i \le n} \neq(b, a_i)$$

which we need to model freshness.

Generic infinite decidable object

U is a 'decidable' object of the topos $[\mathbb{I}, \mathbf{Set}]$

$$a =_{\mathsf{U}} b \ \wedge \ \neq(a,b) \ \Rightarrow \ \mathbf{false}$$
$$\mathbf{true} \ \Rightarrow \ a =_{\mathsf{U}} b \ \vee \ \neq(a,b)$$

but it does not satisfy 'finite inexhaustibility'

$$\bigwedge_{0\le i<j\le n} \neq(a_i, a_j) \ \Rightarrow \ \bigvee_{b:\mathsf{U}} \bigwedge_{0\le i\le n} \neq(b, a_i)$$

FACT: we get this form of infinity (in a geometrically generic way) if we cut down to the Schanuel topos:

$\mathbf{Sch} \subseteq [\mathbb{I}, \mathbf{Set}]$ is the full subcategory consisting of functors $\mathbb{I} \longrightarrow \mathbf{Set}$ that preserve pullbacks

Generic infinite decidable object

$\mathbf{U}$ is a 'decidable' object of the topos $[\mathbb{I}, \mathbf{Set}]$

$$a =_{\mathsf{U}} b \ \wedge\ \neq(a,b) \ \Rightarrow\ \mathbf{false}$$
$$\mathbf{true} \ \Rightarrow\ a =_{\mathsf{U}} b \ \vee\ \neq(a,b)$$

but it does not satisfy 'finite inexhaustibility'

$$\bigwedge_{0 \le i < j \le n} \neq(a_i, a_j) \ \Rightarrow\ \bigvee_{b:\mathsf{U}} \bigwedge_{0 \le i \le n} \neq(b, a_i)$$

FACT: we get this form of infinity (in a geometrically generic way) if we cut down to the Schanuel topos.

What is the history of this notion? (Kuratowski?)

From **Sch** to **Nom**

The category of nominal sets $\boxed{\mathbf{Nom}}$ is 'merely' an equivalent presentation of the category **Sch**:

An analogy:

$$\frac{\mathbf{Nom}}{\mathbf{Sch}} \sim \frac{\text{named bound variables}}{\text{de Bruijn indexes (levels)}}$$

Step 1: fix a countably infinite set $\boxed{\mathbb{A}}$ (of atomic names) and modify **Sch** up to equivalence by replacing $\mathbb{I}$ by the equivalent category whose objects are finite subsets $I \in \mathbf{P}_{\mathbf{fin}}\, \mathbb{A}$ and whose morphisms are injective functions.

From **Sch** to **Nom**

The category of nominal sets $\boxed{\mathbf{Nom}}$ is 'merely' an equivalent presentation of the category **Sch**:

Step 2: make the dependence of each $X \in \mathbf{Sch}$ on 'possible worlds' $A \in \mathbf{P_{fin}}\,\mathbb{A}$ implicit by taking the colimit $\tilde{X}$ of the directed system of sets and (injective) functions

$$A \subseteq B \in \mathbf{P_{fin}}\,\mathbb{A} \;\mapsto\; (X\,A \to X\,B)$$

Each set $\tilde{X}$ carries an action of $\mathbb{A}$-permutations

(cf. homogeneity property (Fraïssé limit) $\boxed{\begin{tikzcd} \mathbb{A} \arrow[r, dashed, "\cong"] & \mathbb{A} \\ A \arrow[u, hook] \arrow[r, tail, "f"'] & B \arrow[u, hook] \end{tikzcd}}$)

From $\mathbf{Sch}$ to $\mathbf{Nom}$

The category of nominal sets $\boxed{\mathbf{Nom}}$ is 'merely' an equivalent presentation of the category $\mathbf{Sch}$:

Step 2: make the dependence of each $X \in \mathbf{Sch}$ on 'possible worlds' $A \in \mathbf{P_{fin}}\,\mathbb{A}$ implicit by taking the colimit $\tilde{X}$ of the directed system of sets and (injective) functions

$$A \subseteq B \in \mathbf{P_{fin}}\,\mathbb{A} \;\mapsto\; (X\,A \to X\,B)$$

Each set $\tilde{X}$ carries an action of $\mathbb{A}$-permutations with finite support property, and every such arises this way up to iso.

Finite support property

Suppose $\mathbf{Perm}\,\mathbb{A}$ (= group of all (finite) permutations of $\mathbb{A}$) acts on a set X and that $x \in X$

A set of names $A \subseteq \mathbb{A}$ supports x if permutations π that fix every $a \in A$ also fix x (i.e. $\pi \cdot x = x$).

X is a nominal set if every $x \in X$ has a finite support.

$\mathbf{Nom}$ = category of nominal sets and functions that preserve the permutation action ($f(\pi \cdot x) = \pi \cdot (f\,x)$).

FACT: $\mathbf{Nom}$ and $\mathbf{Sch}$ are equivalent categories.

Within $\mathbf{Nom}$, objects are 'set-like' and the modal character of freshness becomes implicit...

Finite support property

Suppose $\mathbf{Perm}\,\mathbb{A}$ (= group of all (finite) permutations of $\mathbb{A}$) acts on a set X and that $x \in X$

A set of names $A \subseteq \mathbb{A}$ supports x if permutations π that fix every $a \in A$ also fix x (i.e. $\pi \cdot x = x$).

X is a nominal set if every $x \in X$ has a finite support.

Freshness, nominally, is a binary relation

$$a \mathbin{\#} x \triangleq a \notin A \text{ for some finite } A \text{ supporting } x.$$

'name a is fresh for x'

Finite support property

Suppose $\mathbf{Perm}\,\mathbb{A}$ (= group of all (finite) permutations of $\mathbb{A}$) acts on a set X and that $x \in X$

A set of names $A \subseteq \mathbb{A}$ supports x if permutations π that fix every $a \in A$ also fix x (i.e. $\pi \cdot x = x$).

X is a nominal set if every $x \in X$ has a finite support.

Freshness, nominally, is a binary relation

$$a \,\#\, x \triangleq a \notin A \text{ for some finite } A \text{ supporting } x.$$

satisfying $\boxed{\forall x. \exists a.\ a \,\#\, x}$ (not Skolemizable!)

Name abstraction

Name abstraction

Each $X \in \mathbf{Nom}$ yields a nominal set $\boxed{[\mathbb{A}]X}$ of

name-abstractions $\langle a \rangle x$ are $\sim$-equivalence classes of pairs $(a, x) \in \mathbb{A} \times X$, where

$$(a, x) \sim (a', x') \Leftrightarrow \exists b \# (a, x, a', x') \\ (b\ a) \cdot x = (b\ a') \cdot x'$$

generalizes α-equivalence from sets of syntax to arbitrary nominal sets

the permutation that swaps a and b

Name abstraction

Each $X \in \mathbf{Nom}$ yields a nominal set $\boxed{[\mathbb{A}]X}$ of

name-abstractions $\langle a \rangle x$ are $\sim$-equivalence classes of pairs $(a, x) \in \mathbb{A} \times X$, where

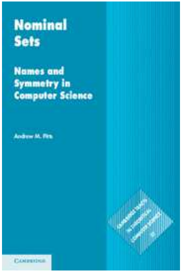
$$(a, x) \sim (a', x') \Leftrightarrow \exists\, b \,\#\, (a, x, a', x') \quad (b\ a) \cdot x = (b\ a') \cdot x'$$

Action of name permutations on $[\mathbb{A}]X$ is well-defined by

$$\pi \cdot \langle a \rangle x = \langle \pi\, a \rangle (\pi \cdot x)$$

and for this action, $A - \{a\}$ supports $\langle a \rangle x$ if A supports x.

If you want to know more about nominal sets…



Nominal Sets

Names and Symmetry in Computer Science

Cambridge Tracts in Theoretical Computer Science, Vol. 57 (CUP, 2013)

Nom and dependent types

Families of nominal sets

Family over $X \in \mathbf{Nom}$ is specified by:

- family of sets $(E_x \mid x \in X)$
- dependently type permutation action

$$\prod_{\pi \in \mathbf{Perm}\,\mathbb{A}} \prod_{x \in X} (E_x \to E_{\pi \cdot x})$$

with dependent version of finite support property:

for all $x \in X, e \in E_x$ there is a finite set A of names supporting x in X and such that any π fixing each $a \in A$ satisfies $\pi \cdot e = e \in E_{\pi \cdot x} = E_x$.

Families of nominal sets

Family over $X \in \mathbf{Nom}$ is specified by...

Get a category with families (cwf) [Dybjer] modelling extensional MLTT...

This cwf is relatively unexplored, so far.

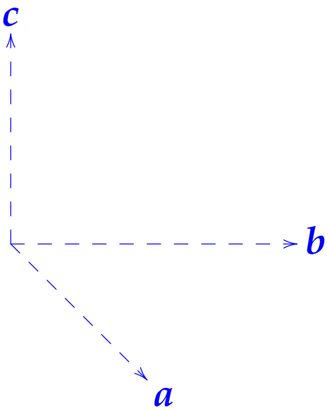
But what's it good for? Two possible applications:

1. higher-dimensional type theory
2. meta-programming/proving with name-binding structures

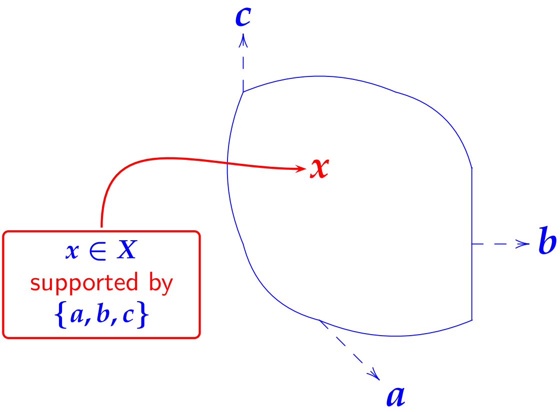
Bezem-Coquand-Huber cubical sets model of HoTT

(just the connection with the nominal sets notion of name abstraction)

One can view cubical sets as nominal sets X equipped with some extra structure, whose names $a, b, c \ldots \in \mathbb{A}$ we think of as names of cartesian directions

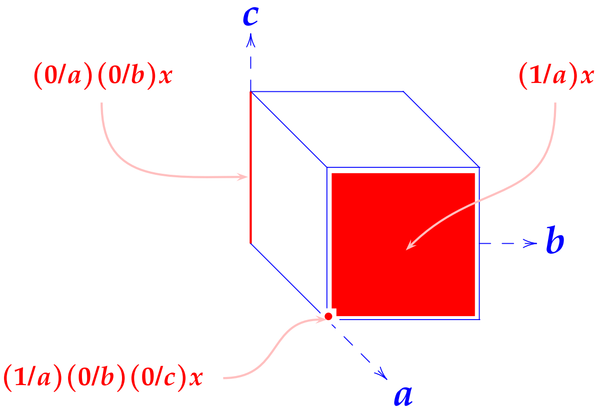


One can view cubical sets as nominal sets X equipped with some extra structure, whose names $a, b, c \ldots \in \mathbb{A}$ we think of as names of cartesian directions



One can view cubical sets as nominal sets X equipped with some extra structure, namely face maps

$$x \in X \mapsto (i/a)x \in X, \text{ for } i = 0, 1$$



One can view cubical sets as nominal sets X equipped with some extra structure,

$$\begin{array}{rcl} d_i : [\mathbb{A}]X & \to & X \\ \langle a \rangle x & \mapsto & (i/a)x \end{array} \qquad (i \in \mathbf{2})$$

satisfying

(binding: $a \# (i/a)x$ – follows from the type of d_i)

degeneracy: $a \# x \Rightarrow (i/a)x = x$

independence: $a \neq b \Rightarrow (i/a)(j/b)x = (j/b)(i/a)x$

$\boxed{\mathbf{Cub}}$ = category of nominal sets equipped with face maps + functions preserving name-permutation action and face maps.

Theorem (Staton). $\mathbf{Cub}$ is equivalent to the presheaf category $[\mathbb{C}, \mathbf{Set}]$ originally used by Bezem, Coquand & Huber.

$\mathbb{C}$ is [equivalent to] the category whose objects are finite ordinals and whose morphisms are given by:

$$\mathbb{C}(m,n) = \{f \in \mathbf{Set}(m+2, n+2) \mid f\,0 = 0 \ \wedge\ f\,1 = 1 \ \wedge \\ \forall i,j > 1.\ f\,i = f\,j > 1 \ \Rightarrow\ i = j\}$$

Name abstractions $\langle a\rangle x$ as paths (proofs of identity) from $(0/a)x$ to $(1/a)x$:

degenerate path
$\mathit{refl}\, x = \langle a\rangle x$
for some/any
$a \mathrel{\#} x$

$$
\begin{tikzcd}
 & {[\mathbb{A}]X} \arrow[d, "{\langle d_0, d_1\rangle}"] \\
X \arrow[ur, "\mathit{refl}"] \arrow[r, "{\langle \mathrm{id},\mathrm{id}\rangle}"'] & X \times X
\end{tikzcd}
\qquad
\begin{tikzcd}
\langle a\rangle x \arrow[d, mapsto] \\
((0/a)x, (1/a)x)
\end{tikzcd}
$$

Can these be the formation and introduction for an (intensional) identity type Id_X for cubical set X?

Name abstractions $\langle a\rangle x$ as paths (proofs of identity) from $(0/a)x$ to $(1/a)x$:

$$\begin{tikzcd} & {[\mathbb{A}]X} \arrow[d, "\langle d_0, d_1\rangle"] & \langle a\rangle x \arrow[d, mapsto] \\ X \arrow[ur, "refl"] \arrow[r, "\langle \mathrm{id}, \mathrm{id}\rangle"'] & X \times X & ((0/a)x, (1/a)x) \end{tikzcd}$$

Can these be the formation and introduction for an (intensional) identity type Id_X for cubical set X?

Bezen-Coquand-Huber: yes (albeit with propositional eliminator), if we take the 'fibrant' families to be given by cubical sets satisfying a uniform Kan filling condition

Name abstractions $\langle a \rangle x$ as paths (proofs of identity) from $(0/a)x$ to $(1/a)x$:

$$\begin{tikzcd} & {[\mathbb{A}]X} \arrow[d, "\langle d_0, d_1 \rangle"] \\ X \arrow[ur, "refl"] \arrow[r, "\langle \mathbf{id}, \mathbf{id} \rangle"'] & X \times X \end{tikzcd} \qquad \begin{tikzcd} \langle a \rangle x \arrow[d, mapsto] \\ ((0/a)x, (1/a)x) \end{tikzcd}$$

Can these be the formation and introduction for an (intensional) identity type Id_X for cubical set X?

Bezen-Coquand-Huber: yes (albeit with propositional eliminator), if we take the 'fibrant' families to be given by cubical sets satisfying a uniform Kan filling condition and one also gets a Voevodsky (univalent) universe.

Name abstractions $\langle a \rangle x$ as paths (proofs of identity) from $(\mathbf{0}/a)x$ to $(\mathbf{1}/a)x$:

$$
\begin{tikzcd}
 & {[\mathbb{A}]X} \arrow[d, "\langle d_0, d_1 \rangle"] \\
X \arrow[ur, "refl"] \arrow[r, "\langle \mathrm{id}, \mathrm{id} \rangle"'] & X \times X
\end{tikzcd}
\qquad
\begin{tikzcd}
\langle a \rangle x \arrow[d, mapsto] \\
((\mathbf{0}/a)x, (\mathbf{1}/a)x)
\end{tikzcd}
$$

Can these be the formation and introduction for an (intensional) identity type Id_X for cubical set X?

Bezen-Coquand-Huber: yes (albeit with propositional eliminator), if we take the 'fibrant' families to be given by cubical sets satisfying a uniform Kan filling condition and one also gets a Voevodsky (univalent) universe.

Why use Kan-$\mathbf{Cub}$ rather than Kan-$[\mathbb{C}, \mathbf{Set}]$?
Variations on Kan filling? 'Nominal' simplicial sets?

Type Theory with names, freshness and name-abstraction

(joint work with Justus Matthiesen)

Families of nominal sets

Family over $X \in \mathbf{Nom}$ is specified by...

Get a category with families (cwf) [Dybjer] modelling extensional MLTT, plus

nominal logic's freshness quantifier	Curry-Howard	dependent name-abstraction
$\text{И}a.\, \varphi(a, \vec{x})$	$\longleftrightarrow$	$[a \in \mathbb{A}]E_a$

Families of nominal sets

Family over $X \in \mathbf{Nom}$ is specified by...

Get a category with families (cwf) [Dybjer] modelling extensional MLTT, plus

nominal logic's freshness quantifier	Curry-Howard	dependent name-abstraction
$\text{И}a.\,\varphi(a,\vec{x})$	$\longleftrightarrow$	$[a \in \mathbb{A}]E_a$

$= \exists a \# \vec{x}.\,\varphi(a,\vec{x})$
$= \forall a \# \vec{x}.\,\varphi(a,\vec{x})$
'some/any fresh a'

Original motivation for Gabbay & AMP to introduce nominal sets and name abstraction:

> $[\mathbb{A}](\_)$ can be combined with $\times$ and $+$ to give functors $\mathbf{Nom} \to \mathbf{Nom}$ that have initial algebras coinciding with sets of abstract syntax trees modulo α-equivalence.

E.g. the initial algebra for $\mathbb{A} + (\_ \times \_) + [\mathbb{A}](\_)$ is isomorphic to the usual set of untyped λ-terms.

Original motivation for Gabbay & AMP to introduce nominal sets and name abstraction...

Initial-algebra universal property $\Rightarrow$ recursion/induction principles for syntax involving name-binding operations [see JACM 53(2006)459-506].

- Exploited in impure functional programming language FreshML [Shinwell, Gabbay & AMP] – recursion only.
- Pure total (recursive) functions and proof (by induction): how to solve the analogy:

$$\frac{\textbf{Coq}}{\textbf{OCaml}} \sim \frac{\textbf{Agda}}{\textbf{Haskell}} \sim \frac{\textbf{?}}{\textbf{FreshML}}$$

Requirements for ‘FreshAgda’

- User-declared sorts of names (possibly with parameters) + user-defined inductive types, with name-abstraction types used to indicate binding constructs. E.g.

```
names  Var : Set

data  Term : Set where
   V : Var -> Term
   A : Term -> Term -> Term
   L : ([Var]Term) -> Term

data  Fresh(X : Set)(x : X) :  Var -> Set where
  fr : [a : Var](Fresh X x a)
```

Requirements for 'FreshAgda'

- User-declared sorts of names (possibly with parameters) + user-defined inductive types, with name-abstraction types used to indicate binding constructs. E.g.

set of λ-terms mod α

```
names Var : Set
 data Term : Set where
     V : Var -> Term
     A : Term -> Term -> Term
     L : ([Var]Term) -> Term
 data Fresh(X : Set)(x : X) : Var -> Set where
    fr : [a : Var](Fresh X x a)
```

set of proofs that a is fresh for `x : X`

Requirements for 'FreshAgda'

- User-declared sorts of names (possibly with parameters) + user-defined inductive types, with name-abstraction types used to indicate binding constructs.
- Extend (dependent) pattern-matching with name-abstraction patterns. E.g.

```
_/_ :  Term -> Var -> Term -> Term
(t/x)(V y)       =  if x == y then t else V y
(t/x)(A t1 t2)   =  A ((t/x)t1) ((t/x )t2)
(t/x)(L <x>t1)   =  L <x>((t/x)t1)
```

capture-avoiding substitution of t for x in t1

Requirements for 'FreshAgda'

- User-declared sorts of names (possibly with parameters) + user-defined inductive types, with name-abstraction types used to indicate binding constructs.
- Extend (dependent) pattern-matching with name-abstraction patterns.

```
_/_ :  Term -> Var -> Term -> Term
(t/x)(V y)        =  if x == y then t else V y
(t/x)(A t1 t2)    =  A ((t/x)t1) ((t/x )t2)
(t/x)(L <x>t1)    =  L <x>((t/x)t1)
```

- Automatically respect α-equivalence:

  FreshML uses impure generativity to ensure this.
  How to do it while maintaining Curry-Howard?

Fact: name abstraction functor

$$[\mathbb{A}](\_) : \mathbf{Nom} \to \mathbf{Nom}$$

is right adjoint to 'separated product' functor

$$(\_) * \mathbb{A} : \mathbf{Nom} \to \mathbf{Nom}$$

where $\boxed{X * \mathbb{A} \triangleq \{(x, a) \mid a \# x\}} \subseteq X \times \mathbb{A}$.

so $[\mathbb{A}]X$ is a kind of (affine) function space
(with a right adjoint!)

$$[\mathbb{A}](\_) : \mathbf{Nom} \to \mathbf{Nom}$$

is right adjoint to 'separated product' functor

$$(\_) * \mathbb{A} : \mathbf{Nom} \to \mathbf{Nom}$$

Counit of the adjunction is 'concretion' of an abstraction

$$\_ @ \_ : ([\mathbb{A}]X) * \mathbb{A} \to X$$

defined by computation rule:

$$(\langle a \rangle x) @ b = (b\ a) \cdot x, \text{ if } b \# \langle a \rangle x$$

Locally fresh names

For example, here are some isomorphisms, described in an informal pseudocode:

$$
\begin{array}{rcl}
i : [\mathbb{A}](X + Y) & \cong & [\mathbb{A}]X + [\mathbb{A}]Y \\
i(z) & = & \textbf{fresh } a \textbf{ in case } z \,@\, a \textbf{ of} \\
 & & \quad \textbf{inl}(x) \to \langle a \rangle x \\
 & & \mid \textbf{inr}(y) \to \langle a \rangle y
\end{array}
$$

Locally fresh names

For example, here are some isomorphisms, described in an informal pseudocode:

$$
\begin{array}{rcl}
\boldsymbol{i} : [\mathbb{A}](X+Y) & \cong & [\mathbb{A}]X + [\mathbb{A}]Y \\
\boldsymbol{i}(z) & = & \textbf{fresh}\ a\ \textbf{in case}\ z \mathbin{@} a\ \textbf{of} \\
 & & \quad \textbf{inl}(x) \to \langle a \rangle x \\
 & & \quad \mid \textbf{inr}(y) \to \langle a \rangle y
\end{array}
$$

given $f \in \mathbf{Nom}(X * \mathbb{A}, Y)$
satisfying $a \# x \Rightarrow a \# f(x, a)$,
we get $\hat{f} \in \mathbf{Nom}(X, Y)$ well-defined by:
$\hat{f}(x) = f(x, a)$ for some/any $a \# x$.
Notation: $\boxed{\textbf{fresh}\ a\ \textbf{in}\ f(x, a)} \triangleq \hat{f}(x)$

Locally fresh names

For example, here are some isomorphisms, described in an informal pseudocode:

$$
\begin{array}{rcl}
i : [\mathbb{A}](X + Y) & \cong & [\mathbb{A}]X + [\mathbb{A}]Y \\
i(z) & = & \textbf{fresh } a \textbf{ in case } z \mathbin{@} a \textbf{ of} \\
 & & \quad \mathbf{inl}(x) \to \langle a \rangle x \\
 & & \mid \mathbf{inr}(y) \to \langle a \rangle y \\
j : ([\mathbb{A}]X \to [\mathbb{A}]Y) & \cong & [\mathbb{A}](X \to Y) \\
j(f) & = & \textbf{fresh } a \textbf{ in} \\
 & & \quad \langle a \rangle (\lambda x.\ f(\langle a \rangle x) \mathbin{@} a)
\end{array}
$$

Can one turn the pseudocode into terms in a formal ‘nominal’ λ-calculus?

Aim: extend (dependently typed) λ-calculus with

names a
name swapping $\texttt{swap}\ a, b\ \texttt{in}\ t$
name abstraction $\langle a \rangle t$ and concretion $t \mathbin{@} a$
locally fresh names $\texttt{fresh}\ \ a\ \texttt{in}\ t$
name equality $\texttt{if}\ \ t = a\ \texttt{then}\ t_1\ \texttt{else}\ t_2$

Aim: extend (dependently typed) λ-calculus with

names a
name swapping $\texttt{swap}\ a, b\ \texttt{in}\ t$
name abstraction $\langle a \rangle t$ and concretion $t \mathbin{@} a$
locally fresh names $\texttt{fresh}\ \ a\ \texttt{in}\ t$
name equality $\texttt{if}\ \ t = a\ \texttt{then}\ t_1\ \texttt{else}\ t_2$

Prior art:

- Stark-Schöpp [CSL 2004] – bunched contexts (+), extensional & undecidable (-)
- Westbrook-Stump-Austin [LFMTP 2009] CNIC – semantics/expressivity?
- Cheney [LMCS 2012] DNTT – bunched contexts (+), no local fresh names (-)
- Crole-Nebel [MFPS 2013] – simple types (-), definitional freshness (+)

Aim: extend (dependently typed) λ-calculus with

names a
name swapping $\mathtt{swap}\ a, b\ \mathtt{in}\ t$
name abstraction $\langle a \rangle t$ and concretion $t \mathbin{@} a$
locally fresh names $\mathtt{fresh}\ \ a\ \mathtt{in}\ t$
name equality $\mathtt{if}\ \ t = a\ \mathtt{then}\ t_1\ \mathtt{else}\ t_2$

Prior art:

- Stark-Schöpp [CSL 2004] – bunched contexts (+), extensional & undecidable (-)
- Westbrook-Stump-Austin [LFMTP 2009] CNIC – semantics/expressivity?
- Cheney [LMCS 2012] DNTT – bunched contexts (+), no local fresh names (-)
- Crole-Nebel [MFPS 2013] – simple types (-), definitional freshness (+)

We cherry pick, aiming for user-friendliness.

Aim: extend (dependently typed) λ-calculus with

names a
name swapping $\mathtt{swap}\ a, b\ \mathtt{in}\ t$
name abstraction $\langle a \rangle t$ and concretion $t \mathbin{@} a$
locally fresh names $\mathtt{fresh}\ a\ \mathtt{in}\ t$
name equality $\mathtt{if}\ t = a\ \mathtt{then}\ t_1\ \mathtt{else}\ t_2$

Difficulty: concretion and locally fresh names are partially defined – have to check freshness conditions.

e.g. for $\textbf{fresh}\ a\ \textbf{in}\ f(x, a)$
to be well-defined, we need
$\boxed{a \mathbin{\#} x \;\Rightarrow\; a \mathbin{\#} f(x, a)}$

Definitional freshness

In a nominal set of (higher-order) functions, proving $a \# f$ can be tricky (undecidable). Common proof pattern:

> Given $a, f, \ldots$, pick a fresh name b and prove $(a\ b) \cdot f = f$. (For functions, equivalent to proving $\forall x.\ (a\ b) \cdot f(x) = f((a\ b) \cdot x)$.)

Definitional freshness

In a nominal set of (higher-order) functions, proving $a \# f$ can be tricky (undecidable). Common proof pattern:

> Given $a, f, \ldots$, pick a fresh name b and prove $(a\ b) \cdot f = f$.
> Since by choice of b we have $b \# f$, we also get $a = (a\ b) \cdot b \# (a\ b) \cdot f = f$, QED.

Definitional freshness

In a nominal set of (higher-order) functions, proving $a \# f$ can be tricky (undecidable). Common proof pattern:

$$\frac{\Gamma \vdash a \# T \qquad \Gamma \vdash t : T \qquad \Gamma\#(b : \mathbb{A}) \vdash (\mathtt{swap}\ a, b\ \mathtt{in}\ t) = t : T}{\Gamma \vdash a \# t : T}$$

bunched contexts, generated by

$$\begin{array}{ccc} \Gamma & \mapsto & \Gamma(x : T) \\ \Gamma & \mapsto & \Gamma\#(a : \mathbb{A}) \end{array}$$

definitional freshness

definitional equality

Definitional freshness

In a nominal set of (higher-order) functions, proving $a \# f$ can be tricky (undecidable). Common proof pattern:

$$\frac{\Gamma \vdash a \# T \qquad \Gamma \vdash t : T \qquad \Gamma\#(b : \mathbb{A}) \vdash (\mathtt{swap}\ a, b\ \mathtt{in}\ t) = t : T}{\Gamma \vdash a \# t : T}$$

Freshness info in bunched contexts gets used via:

$$\frac{\Gamma(x : T)\Gamma'\ \mathbf{ok} \qquad a, b \in \Gamma'}{\Gamma(x : T)\Gamma' \vdash (\mathtt{swap}\ a, b\ \mathtt{in}\ x) = x : T}$$

Definitional freshness

In a nominal set of (higher-order) functions, proving $a \# f$ can be tricky (undecidable). Common proof pattern:

$$\frac{\Gamma \vdash a \# T \qquad \Gamma \vdash t : T \qquad \Gamma\#(b : \mathbb{A}) \vdash (\mathtt{swap}\ a, b\ \mathtt{in}\ t) = t : T}{\Gamma \vdash a \# t : T}$$

definitional freshness for types:

$$\frac{\Gamma \vdash T \qquad a \in \Gamma \qquad \Gamma\#(b : \mathbb{A}) \vdash (\mathtt{swap}\ a, b\ \mathtt{in}\ T) = T}{\Gamma \vdash a \# T}$$

A type theory

OMITTED

$\boxed{\Gamma \vdash}$

$$\frac{\Gamma \vdash \quad a \notin \mathrm{dom}\,\Gamma}{\Gamma(\# a) \vdash}\ \text{(ATM-HYP)}$$

$\boxed{\Gamma \vdash T}$

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{Atm}}\ \text{(ATM-FORM)} \qquad \frac{\Gamma(\# a) \vdash T}{\Gamma \vdash (\# a){\to}T}\ \text{(ABS-FORM)} \qquad \frac{\Gamma(\# a) \vdash a \# T}{\Gamma \vdash \cdot}$$

$\boxed{\Gamma \vdash a \# T}$

$$\frac{a \in \mathrm{dom}\,\Gamma \quad \Gamma \vdash T \quad \Gamma(\# a')}{\Gamma \vdash a \#}$$

$\boxed{\Gamma \vdash T = T'}$

$\boxed{\Gamma \vdash}$

(...INTRO)

$$\frac{\# (e : T)}{va.e : va.T}\ \text{(LOCAL-INTRO)}$$

$$\frac{\Gamma \vdash a' \# (e : (\# a){\to}T)}{\Gamma \vdash e \,@\, a' : va.(a\ a') * T}\ \text{(ABS-ELIM)}$$

(...EF-FRESH-2)

$$\frac{(\# a)\Gamma' \vdash e = e' : T}{e' : T}\ \text{(ATM-STRENGTHEN)}$$

$$\frac{)\Gamma' \vdash \quad a, a' \in \mathrm{dom}\,\Gamma'}{\Gamma(x : T)\Gamma' \vdash (a\ a') * x = x : T}\ \text{(SWAP-FRESH-VAR)}$$

$$\frac{a \in \mathrm{dom}_{\mathbb{A}}\Gamma \quad \Gamma \vdash e_1 : T \quad \Gamma \vdash e_2 : T}{\Gamma \vdash (\mathsf{if}\ a = a\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2) = e_1 : T}\ \text{(IF-COMP-1)}$$

$$\frac{\Gamma \vdash a \# (e : \mathsf{Atm}) \quad \Gamma \vdash e_1 : T \quad \Gamma \vdash e_2 : T}{\Gamma \vdash (\mathsf{if}\ e = a\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2) = e_2 : T}\ \text{(IF-COMP-2)}$$

$$\frac{\Gamma(\# a) \vdash a \# (e : T) \quad \Gamma(\# a)\Gamma' \vdash}{\Gamma(\# a)\Gamma' \vdash va.e = e : T}\ \text{(LOCAL-COMP-2)}$$

$$\frac{\Gamma(\# a) \vdash e : T \quad \Gamma(\# a) \vdash a' \# (e : T) \quad a \neq a'}{\Gamma \vdash (\alpha(\# a){\to}e) \,@\, a' = va.(a\ a') * e : va.(a\ a') * T}\ \text{(ABS-COMP)}$$

$$\frac{\Gamma \vdash e : (\# a){\to}T}{\Gamma \vdash e = \alpha(\# a){\to}(e \,@\, a) : (\# a){\to}T}\ \text{(ABS-UNIQ)}$$

To do

- Decidability of typing & definitional equality judgements (normal forms and algorithmic version of the type system).
- Inductively defined types involving $[a : \mathbb{A}](\_)$ (e.g. propositional freshness & nominal logic).
- Dependently typed pattern-matching with name-abstraction patterns.
- Implementation.

Conclusions

1. $\mathbf{Nom}$ *vs* $\mathbf{Sch}$, $\mathbf{Cub}$ *vs* $[\mathbb{C}, \mathbf{Set}]$: names are convenient! (because unlike indexes, they survive weakening).
2. Possibility of a 'nominal' treatment of dimensions in higher-dimensional type theory & category theory seems intriguing: e.g. what are ∞-groupoids when $\infty =$ finitely inexhaustible?
3. Nominal sets notion of implicit dependence does not sit easily with explicit functional dependence in type theory. (Permutations are mathematically pleasant, but not computationally pleasant?)