Semantics of Local Names

Andrew Pitts

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

MFPS XXXI & CALCO 2015

nominal techniques • algebraic effects

Local names

- **Local variables** in Algol-like languages:
  $\mathtt{new}\ X\ \mathtt{in}\ \langle command \rangle$

- **Generativity** + local declarations in ML-like languages:
  $\mathtt{let}\ x = \mathtt{ref}\langle val \rangle\ \mathtt{in}\ \langle exp \rangle$

- Channel-**name restriction** in π-like process calculi:
  $(\nu a)\langle process \rangle$

- Use of **fresh names** in meta-programming/reasoning, e.g.

$$\text{A-nf}(e_1\, e_2) \triangleq \mathtt{let}\ v_1 = e_1, v_2 = e_2\ \mathtt{in}\ v_1\, v_2$$

where $v_1\ v_2$ *are fresh*

Local names

- **Local variables** in Algol-like languages:
  $\texttt{new}\ X\ \texttt{in}\ \langle command \rangle$
- **Generativity** + local declarations in ML-like languages:
  $\texttt{let}\ x = \texttt{ref}\,\langle val \rangle\ \texttt{in}\ \langle exp \rangle$
- Channel-**name restriction** in π-like process calculi:
  $(\nu a)\langle process \rangle$
- Use of **fresh names** in meta-programming/reasoning, e.g.

$$\text{A-nf}(e_1\, e_2) \triangleq \texttt{let}\ v_1 = e_1,\ v_2 = e_2\ \texttt{in}\ v_1\, v_2$$

where $v_1\ v_2$ *are fresh*

What is the mathematical foundation for these locality constructs? Is it the same in each case?

What is the mathematical foundation for these locality constructs? Is it the same in each case?

I've had a 20+ year interest (obsession?) in such questions, aided by

James Cheney, Ranald Clouston, Roy Crole, Jasper Derikx, Dan Ghica, Marcelo Fiore, Jamie Gabbay, Matthew Hennessy, Matt Lakin, Steffen Lösch, Justus Matthiesen, Frank Nebel, Eike Ritter, Uli Schöpp, Mark Shinwell, Ian Stark, Sam Staton, Alley Stoughton, Christian Urban, ...

What is the mathematical foundation for these locality constructs? Is it the same in each case?

I've had a 20+ year interest (obsession?) in such questions, aided by

James Cheney, Ranald Clouston, Roy Crole, Jasper Derikx, Dan Ghica, Marcelo Fiore, Jamie Gabbay, Matthew Hennessy, Matt Lakin, Steffen Lösch, Justus Matthiesen, Frank Nebel, Eike Ritter, Uli Schöpp, Mark Shinwell, Ian Stark, Sam Staton, Alley Stoughton, Christian Urban, ...

What's new?

What is the mathematical foundation for these locality constructs? Is it the same in each case?

What's new:

Integration with dependent type theory (DTT)

Motivation: programming or proving?

What is the mathematical foundation for these locality constructs? Is it the same in each case?

What's new:

Integration with dependent type theory (DTT)

Motivation: programming or proving.

- DTT with generative local names
- Nominal techniques
- Judgemental freshness

DTT with generative names

DTT judgements $\begin{cases} \text{typing} & T\ \textbf{type} \quad t : T \\ \text{equality} & T = T' \quad t = t' : T \end{cases}$

are intertwined:

$$\frac{(x : T_1) \vdash T_2(x)\ \textbf{type} \qquad t = t' : T_1}{T_2(t) = T_2(t')}$$

$$\frac{t : T_1 \qquad T_1 = T_2}{t : T_2}$$

DTT judgements $\begin{cases} \text{typing} & T \textbf{ type} \quad t : T \\ \text{equality} & T = T' \quad t = t' : T \end{cases}$

If we allow generative, locally scoped names $\nu x.\, t(x)$, what are the rules for (decidable) equality judgements?

DTT judgements $\begin{cases} \text{typing} & T \textbf{ type} \quad t : T \\ \text{equality} & T = T' \quad t = t' : T \end{cases}$

If we allow generative, locally scoped names $\nu x.\, t(x)$, what are the rules for (decidable) equality judgements?

Familiar generative dynamics of locally scoped names

$$(\nu x.\, t(x)\ ,\, state) \to (t(a)\ ,\, state \uplus \{a\})$$

DTT judgements $\begin{cases} \text{typing} & T\ \textbf{type} \quad t : T \\ \text{equality} & T = T' \quad t = t' : T \end{cases}$

If we allow generative, locally scoped names $\nu x.\, t(x)$,
what are the rules for (decidable) equality judgements?

Familiar generative dynamics of locally scoped names

$$(\nu x.\, t(x)\ , state) \to (t(a)\ , state \uplus \{a\})$$

can be reformulated with evaluation contexts $E[\_]$

$$E[\nu x.\, t(x)] \to \nu x.\, E[t(x)]$$

to answer this question.

DTT judgements $\begin{cases} \text{typing} & T\ \mathbf{type} & t : T \\ \text{equality} & T = T' & t = t' : T \end{cases}$

If we allow generative, locally scoped names $\nu x.\, t(x)$, what are the rules for (decidable) typing judgements?

Typing generative local names

$$\frac{T\ \textbf{type} \qquad (x : \mathit{Name}) \vdash t(x) : T}{\nu x.\, t(x) : T}$$

is safe, but inexpressive – seems inevitable that type expressions as well as term expressions may involve name generation:

$$\frac{(x : \mathit{Name}) \vdash t(x) : T(x)}{\nu x.\, t(x) : \nu x.\, T(x)}$$

Typing generative local names

$$\frac{T\ \mathbf{type} \qquad (x : Name) \vdash t(x) : T}{\nu x.\, t(x) : T}$$

is safe, but inexpressive – seems inevitable that type expressions as well as term expressions may involve name generation:

$$\frac{(x : Name) \vdash t(x) : T(x)}{\nu y.\, t(y) : \nu z.\, T(z)}$$

If t & T are both generative, what does $t : T$ mean? Are there models to guide us?

Nominal techniques

Nominal sets overview

Fix countably infinite set $\mathbb{A}$ (elements $a, b, c, \ldots$ called atoms).

Nominal set =

$$\text{set } D + \left(\begin{array}{c} \text{atom-swapping function} \\ (\_\ \_)\cdot\_ : \mathbb{A} \to \mathbb{A} \to D \to D \\ \text{with simple algebraic properties} \end{array} \right) + \text{finite supports}$$

Morphism of nominal sets =

function that commutes with atom swapping

Nominal sets overview

Fix countably infinite set $\mathbb{A}$ (elements $a, b, c, \ldots$ called atoms).

Nominal set =

set D + (atom-swapping function $(\_\ \_)\cdot\_ : \mathbb{A} \to \mathbb{A} \to D \to D$ with simple algebraic properties) + finite supports

Morphism of nominal sets =

function that commutes with atom swapping

for every $d \in D$ there is
a finite list of names $S_d \in \mathbf{List}\,\mathbb{A}$
satisfying $(\forall a, b \notin S_d)\ (a\ b) \cdot d = d$

(S_d lists the names that d *may involve*.
In this talk I will try to be constructive,
so no use of *least* support sets.)

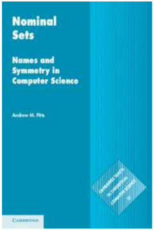
Nominal sets overview

Fix countably infinite set $\mathbb{A}$ (elements $a, b, c, \ldots$ called atoms).

Nominal set =

$$\text{set } D + \left(\begin{array}{c} \text{atom-swapping function} \\ (\_\ \_)\cdot\_ : \mathbb{A} \to \mathbb{A} \to D \to D \\ \text{with simple algebraic properties} \end{array} \right) + \text{finite supports}$$

Morphism of nominal sets =
function that commutes with atom swapping

If you want to know more, then read 

Families of nominal sets

Equivalent presentation of slice categories $\mathbf{Nom}/D$ making pullbacks associate 'on the nose':

Families of nominal sets

Equivalent presentation of slice categories $\mathbf{Nom}/D$ making pullbacks associate 'on the nose':

A family over $D \in \mathbf{Nom}$ is specified by:

- D-indexed family of sets $(E_d \mid d \in D)$
- dependently typed atom-swapping

$$(a\ b) \cdot \_ \; : \; E_d \to E_{(a\ b) \cdot d}$$

with dependent version of finite support property.

Get a category with families (CwF) [Dybjer, 1996] modelling extensional MLTT...

DTT	CwF **Nom**
contexts	objects
$\mathbf{\Gamma} \vdash$	$D \in \mathbf{Nom}$
types	families
$\mathbf{\Gamma} \vdash T\ \mathbf{type}$	$\left(\begin{array}{c}\sum_{d\in D} E_d \\ \downarrow \\ \mathbf{\Gamma}\end{array}\right) \in \mathbf{Nom}/D$
terms	global sections
$\mathbf{\Gamma} \vdash t : T$	$\left(\begin{array}{ccc} D & \longrightarrow & \sum_{d\in D} E_d \\ & \mathbf{id} \searrow & \downarrow \\ & & D\end{array}\right) \in \mathbf{Nom}/D$

See [M. Hofmann, *Syntax and Semantics of Dependent Types*, 1997].

Judgemental freshness

Local names

- **Local variables** in Algol-like languages:
  `new` X `in` $\langle command \rangle$
- **Generativity** + local declarations in ML-like languages:
  `let` $x =$ `ref` $\langle val \rangle$ `in` $\langle exp \rangle$
- Channel-**name restriction** in π-like process calculi:
  $(\nu a)\langle process \rangle$
- Use of **fresh names** in meta-programming/reasoning, e.g.

$$\text{A-nf}(e_1\, e_2) \triangleq \texttt{let } v_1 = e_1, v_2 = e_2 \texttt{ in } v_1\, v_2$$

where $v_1\, v_2$ *are fresh*

Thesis: fresh names in metaprogramming/reasoning are always used in way that is semantically trivial.

Semantic freshness

freshness relation $\_\,\#\,\_ \subseteq \mathbb{A} \times D$ (for $D \in \mathbf{Nom}$)

$$\begin{aligned} a \mathbin{\#} d \;&\triangleq\; (\exists b \notin S_d)\; (a\; b) \cdot d = d \\ &\Leftrightarrow\; (\forall b \notin S_d)\; (a\; b) \cdot d = d \end{aligned}$$

provides a syntax-independent notion of

freeness/non-occurrence

Semantic freshness

freshness relation $\_ \# \_ \subseteq \mathbb{A} \times D$ (for $D \in \mathbf{Nom}$)

$$a \# d \triangleq (\exists b \notin S_d)\ (a\ b) \cdot d = d$$
$$\Leftrightarrow (\forall b \notin S_d)\ (a\ b) \cdot d = d$$

If d is described by a term t with no free variables, then $\{$free atoms of $t\}$ will do for S_d.

Semantic freshness

freshness relation $\_ \# \_ \subseteq \mathbb{A} \times D$ (for $D \in \mathbf{Nom}$)

$$
\begin{aligned}
a \# d \;&\triangleq\; (\exists b \notin S_d)\ (a\ b) \cdot d = d \\
&\Leftrightarrow\; (\forall b \notin S_d)\ (a\ b) \cdot d = d
\end{aligned}
$$

If d is described by a term t with no free variables, then {free atoms of t} will do for S_d. But what if t does have free variables?

DTT	CwF **Nom**
extend context with a variable $$\frac{\boldsymbol{\Gamma \vdash T\ \mathbf{type}}}{\boldsymbol{\Gamma(x : T) \vdash}}(x \notin \Gamma)$$	dependent product $$\frac{(E_d \mid d \in D)}{\sum_{d \in D} E_d}$$

Bunched contexts

DTT+names	CwF $\mathbf{Nom}$
extend context with a variable $$\frac{\Gamma \vdash T\ \mathbf{type}}{\Gamma(x : T) \vdash}\,(x \notin \Gamma)$$	dependent product $$\frac{(E_d \mid d \in D)}{\sum_{d \in D} E_d}$$
extend context with a fresh name $$\frac{\Gamma \vdash}{\Gamma[a : Name] \vdash}\,(a \notin \Gamma)$$	separated product $$\frac{D}{D \otimes \mathbb{A}}$$

In DTT+names:
$Name$ is a type of names,
$[\![Name]\!] = \mathbb{A}$ (nominal set of atoms),
variables x and *atoms* a are disjoint classes of identifier.

Bunched contexts

DTT+names	CwF **Nom**
extend context with a variable	dependent product
$\dfrac{\Gamma \vdash T\ \mathbf{type}}{\Gamma(x : T) \vdash}(x \notin \Gamma)$	$\dfrac{(E_d \mid d \in D)}{\sum_{d \in D} E_d}$
extend context with a fresh name	separated product
$\dfrac{\Gamma \vdash}{\Gamma[a : Name] \vdash}(a \notin \Gamma)$	$\dfrac{D}{D \otimes \mathbb{A}}$

$D \otimes \mathbb{A}$: $\{(d, a) \in D \times \mathbb{A} \mid a \# d\}$

See [Stark-Schöpp, CSL 2004][Cheney, LMCS 2012].

Judgemental freshness

Judgemental freshness is derivable from judgemental equality

cf. [Clouston, LFMTP 2011] and [Crole-Nebel, MFPS 2013]

$$\frac{\Gamma \vdash a : Name \qquad \Gamma \vdash t : T \qquad \Gamma[b : Name] \vdash (\mathtt{swap}\, a\, , b\, \mathtt{in}\, t) = t : T}{\Gamma \vdash a \,\#\, t : T}\ (b \notin \Gamma)$$

($\mathtt{swap}\, a\, , b\, \mathtt{in}\, t$ is an *explicit swapping* expression)

Judgemental freshness

Judgemental freshness is derivable from judgemental equality

cf. [Clouston, LFMTP 2011] and [Crole-Nebel, MFPS 2013]

$$\frac{\Gamma \vdash a \# T \qquad \Gamma \vdash t : T \qquad \Gamma[b : Name] \vdash (\mathtt{swap}\, a, b\, \mathtt{in}\, t) = t : T}{\Gamma \vdash a \# t : T}\ (b \notin \Gamma)$$

$$\frac{\Gamma \vdash a : Name \qquad \Gamma \vdash T \qquad \Gamma[b : Name] \vdash (\mathtt{swap}\, a, b\, \mathtt{in}\, T) = T}{\Gamma \vdash a \# T}\ (b \notin \Gamma)$$

syntactic	$\subseteq$	judgemental	$\subseteq$	semantic
freshness	$\neq$	freshness	$\neq$	freshness

syntactic freshness	$\subset$ $\neq$	judgemental freshness	$\subset$ $\neq$	semantic freshness

a occurs in $\texttt{if}\ a = b\ \texttt{then}\ a\ \texttt{else}\ b$, but
$[a\ b : \mathit{Name}] \vdash a \mathbin{\#} (\texttt{if}\ a = b\ \texttt{then}\ a\ \texttt{else}\ b) : \mathit{Name}$

syntactic freshness $\subset$ judgemental freshness $\subset$ semantic freshness

syntactic freshness $\neq$ judgemental freshness $\neq$ semantic freshness

a occurs in $\texttt{if}\ a = b\ \texttt{then}\ a\ \texttt{else}\ b$, but
$[a\ b : \mathit{Name}] \vdash a \mathbin{\#} (\texttt{if}\ a = b\ \texttt{then}\ a\ \texttt{else}\ b) : \mathit{Name}$

RHS is not in general a decidable relation, but (conjecture) the LHS is.

Local names

- **Local variables** in Algol-like languages:
  `new` X `in` $\langle command \rangle$
- **Generativity** + local declarations in ML-like languages:
  `let` $x =$ `ref` $\langle val \rangle$ `in` $\langle exp \rangle$
- Channel-**name restriction** in π-like process calculi:
  $(\nu a)\langle process \rangle$
- Use of **fresh names** in meta-programming/reasoning, e.g.

$$\text{A-nf}(e_1\, e_2) \triangleq \texttt{let } v_1 = e_1, v_2 = e_2 \texttt{ in } v_1\, v_2$$

where $v_1\, v_2$ *are fresh*

Thesis: fresh names in metaprogramming/reasoning are always used in way that is semantically trivial.

Freshness theorem for $\mathbf{Nom}$

If $f \in \mathbf{Nom}(\mathbb{A} \times D, D')$ satisfies for all a,d

$$a \# d \Rightarrow a \# f(a,d)$$

then $\exists$ unique $f' \in \mathbf{Nom}(D, D')$ s.t. for all a,d

$$a \# d \Rightarrow f'd = f(a,d)$$

(so $f'd$ is $f(a,d)$ for some/any fresh a)

Freshness theorem for **Nom**

If $f \in \mathbf{Nom}(\mathbb{A} \times D, D')$ satisfies for all a, d

$$a \mathbin{\#} d \;\Rightarrow\; a \mathbin{\#} f(a,d)$$

then $\exists$ unique $f' \in \mathbf{Nom}(D, D')$ s.t. for all a, d

$$a \mathbin{\#} d \;\Rightarrow\; f'd = f(a,d)$$

(so $f'd$ is $f(a,d)$ for some/any fresh a)

We can express this kind of
semantically trivial locally scoped name
in DTT + judgemental freshness, replacing this
with $\Gamma[a : \mathit{Name}] \vdash a \mathbin{\#} t : T$
and introducing syntax for f' as a function of f...

Judgementally fresh locally scoped names

Formation and introduction:

$$\frac{\Gamma[a : \mathbb{A}] \vdash a \# T(a)}{\Gamma \vdash \nu b.\, T(b)} \qquad \frac{\Gamma[a : \mathbb{A}] \vdash a \# t(a) : T(a)}{\Gamma \vdash \nu b.\, t(b) : \nu c.\, T(c)}$$

Judgementally fresh locally scoped names

Formation and introduction:

$$\frac{\Gamma[a : \mathbb{A}] \vdash a \mathbin{\#} T(a)}{\Gamma \vdash \nu b.\, T(b)} \qquad \frac{\Gamma[a : \mathbb{A}] \vdash a \mathbin{\#} t(a) : T(a)}{\Gamma \vdash \nu b.\, t(b) : \nu c.\, T(c)}$$

Computationally, $\nu a.\, \_$ is a no-op...

$$\frac{\Gamma[a : \mathbb{A}] \vdash a \mathbin{\#} T(a)}{\Gamma[a : \mathbb{A}] \vdash \nu b.\, T(b) = T(a)}$$

$$\frac{\Gamma[a : \mathbb{A}] \vdash a \mathbin{\#} t(a) : T(a)}{\Gamma[a : \mathbb{A}] \vdash \nu b.\, t(b) = t(a) : T(a)}$$

Judgementally fresh locally scoped names

Formation and introduction:

$$\frac{\Gamma[a : \mathbb{A}] \vdash a \mathrel{\#} T(a)}{\Gamma \vdash \nu b.\, T(b)} \qquad \frac{\Gamma[a : \mathbb{A}] \vdash a \mathrel{\#} t(a) : T(a)}{\Gamma \vdash \nu b.\, t(b) : \nu c.\, T(c)}$$

Computationally, $\nu a.\,\_$ is a no-op...

$$\frac{\Gamma[a : \mathbb{A}] \vdash a \mathrel{\#} T(a)}{\Gamma[a : \mathbb{A}] \vdash \nu b.\, T(b) = T(a)}$$

$$\frac{\Gamma[a : \mathbb{A}] \vdash a \mathrel{\#} t(a) : T(a)}{\Gamma[a : \mathbb{A}] \vdash \nu b.\, t(b) = t(a) : T(a)}$$

Sound interpretation in the CwF **Nom** using the Freshness Theorem.

FreshMLTT

[AMP, J. Matthiesen and J. Derikx, *A Dependent Type Theory with Abstractable Names*, LSFA 2014.]

- intensional Martin-Löf Type Theory
  + swappable names
  + judgementally fresh, locally scoped names
  + (dependent) name-abstraction types.
- Sound semantics using the CwF of nominal sets.
- Prototype implementation in development by Matthiesen.

FreshMLTT

[AMP, J. Matthiesen and J. Derikx, *A Dependent Type Theory with Abstractable Names*, LSFA 2014.]

- intensional Martin-Löf Type Theory
  $+$ swappable names
  $+$ judgementally fresh, locally scoped names
  $+$ (dependent) name-abstraction types.
- Sound semantics using the CwF of nominal sets.
- Prototype implementation in development by Matthiesen.

To do: FreshMLTT has interesting (?) new forms of inductively defined indexed families of types using constructors with dependent name-abstractions in their arities (e.g. *propositional freshness* type – Curry-Howard for nominal logic).

Typing generative local names

$$\frac{T\ \textbf{type} \qquad (x : Name) \vdash t(x) : T}{\nu x.\, t(x) : T}$$

is safe, but inexpressive – seems inevitable that type expressions as well as term expressions may involve name generation:

$$\frac{(x : Name) \vdash t(x) : T(x)}{\nu y.\, t(y) : \nu z.\, T(z)}$$

If t & T are both generative, what does $t : T$ mean? Are there models to guide us?

Typing generative local names

Wanted: a design combining DTT with
generative locally scoped names
that is user-friendly (no monadic-style
over-sequentialization of the effect of name creation)
and with a simple semantic model.

If you have one, see me afterwards!

If t & T are both generative, what does $t : T$ mean?
Are there models to guide us?

END