Semantics of Locally Scoped Names

Andrew Pitts (joint work with Steffen Lösch)
Happy birthday!

[ PI E R R E - L O U I S C U R I E N ]
Happy birthday!

[ Pierre Pierre - Louis Categorie ]
Happy birthday!

[ P I E R R E - ]

[ L O G I C ]

Curien-fest 2013
Happy birthday!

P R O G R A M M I N G

L O C A T I O N S
\( \nu\text{PCF} = \) higher-order computable functions (PCF) + locally scoped names (Odersky-style).

Example \( \nu\text{PCF} \) expressions:

\[
eq_a \triangleq \lambda x : \text{name}. \quad \text{if } x = a \text{ then } T \text{ else } F
\]

\[
F_1 \triangleq \lambda q : (\text{name} \rightarrow \text{bool}) \rightarrow \text{bool}. \quad \nu a. (q \; \eq_a)
\]

\( \eq_a \) has type \( \text{name} \rightarrow \text{bool} \).

\( F_1 \) has type \( ((\text{name} \rightarrow \text{bool}) \rightarrow \text{bool}) \rightarrow \text{bool} \).
νPCF = higher-order computable functions (PCF) + locally scoped names (Odersky-style).

- New (computationally adequate) denotational semantics, using ‘symmetry-aware’ Scott domains.
\( \nu PCF = \) higher-order computable functions (PCF) + locally scoped names (Odersky-style).

- New (computationally adequate) denotational semantics, using ‘symmetry-aware’ Scott domains.
- Unlike for PCF, semantics is not fully abstract for \( \nu PCF + \) parallel or
\[ \nu \text{PCF} = \text{higher-order computable functions (PCF)} + \text{locally scoped names (Odersky-style)}. \]

- New (computationally adequate) denotational semantics, using \textit{‘symmetry-aware’ Scott domains}.
- Unlike for PCF, semantics is not fully abstract for \( \nu \text{PCF} + \text{parallel or} \)
- But is fully abstract for \( \nu \text{PCF} + \text{parallel or} + \text{exists name} + \text{the name} \).
Finite support example

- Flat domains $A \perp$, $2\perp$, where
  $A = \{a_0, a_1, \ldots\}$ infinite set of ‘symbols’
  $2 = \{0, 1\}$ booleans

- Existential quantifier $f \in (A \perp \rightarrow 2\perp) \mapsto \exists f \in 2\perp$

  $\exists f \triangleq \begin{cases} 
  1 & \text{if } (\exists a \in A) f(a) = 1 \\
  0 & \text{if } (\forall a \in A) f(a) = 0 \\
  \bot & \text{otherwise}
  \end{cases}$
Finite support example

- Flat domains $\mathbb{A}_\perp$, $2_\perp$, where
  $\mathbb{A} = \{a_0, a_1, \ldots\}$ infinite set of ‘symbols’
  $2 = \{0, 1\}$ booleans

- Existential quantifier $f \in (\mathbb{A}_\perp \rightarrow 2_\perp) \mapsto \exists f \in 2_\perp$
  
  \[
  \exists f \triangleq \begin{cases}
  1 & \text{if } (\exists a \in \mathbb{A}) f(a) = 1 \\
  0 & \text{if } (\forall a \in \mathbb{A}) f(a) = 0 \\
  \bot & \text{otherwise}
  \end{cases}
  \]

  does not give a continuous function $\exists : (\mathbb{A}_\perp \rightarrow 2_\perp) \rightarrow 2_\perp$

- e.g. consider limit of $f_n : a_i \mapsto \begin{cases}
  0 & i < n \\
  \bot & i \geq n
  \end{cases}$
Finite support example

- Flat domains $\mathbb{A}_\bot$, $\mathbb{2}_\bot$, where
  $\mathbb{A} = \{a_0, a_1, \ldots\}$ infinite set of ‘symbols’
  $\mathbb{2} = \{0, 1\}$ booleans

- Existential quantifier $f \in (\mathbb{A}_\bot \rightarrow \mathbb{2}_\bot) \mapsto \text{exists } f \in \mathbb{2}_\bot$

  \[
  \exists f \triangleq \begin{cases}
  1 & \text{if } (\exists a \in \mathbb{A}) f\ a = 1 \\
  0 & \text{if } (\forall a \in \mathbb{A}) f\ a = 0 \\
  \bot & \text{otherwise}
  \end{cases}
  \]

  does not give a continuous function $\exists f : (\mathbb{A}_\bot \rightarrow \mathbb{2}_\bot) \rightarrow \mathbb{2}_\bot$
  but it does when restricted to finitely supported functions...
Finite support

- Infinitely many symbols $a \in A$.
- Sets equipped with an action $\pi, x \mapsto \pi \cdot x$ of (finite) permutations $\pi : A \cong A$:

\[
\begin{align*}
\text{id} \cdot x &= x \\
\pi' \cdot (\pi \cdot x) &= (\pi' \circ \pi) \cdot x
\end{align*}
\]
Finite support

- Infinitely many symbols $a \in \mathcal{A}$.
- **Nominal set** $= \text{set } D$ equipped with a permutation action for which each $x \in D$ possess a finite support $A \subseteq \mathcal{A}$:

\[
(\forall \pi) \left( (\forall a \in A) \pi a = a \right) \Rightarrow \pi \cdot x = x
\]

if there is such an $A$, there’s a least one, written $\text{supp } x$
Finite support

- Infinitely many symbols $a \in A$.
- Nominal set = set $D$ equipped with a permutation action for which each $x \in D$ possess a finite support.
- Category of nominal sets & action-preserving functions is a well-known (2-valued) topos.

Exponentials: $D \rightarrow_{fs} E \triangleq$ all functions $f : D \rightarrow E$ that are finitely supported w.r.t. action $\pi \cdot f : x \mapsto \pi \cdot (f(\pi^{-1} \cdot x))$.

Group inverses make exponentials much simpler than for more general monoid/category actions.
Finite support example

- Flat domains $\mathbb{A}_\bot$, $2_\bot$, where
  $\mathbb{A} = \{a_0, a_1, \ldots\}$ infinite set of ‘symbols’
  $2 = \{0, 1\}$ booleans

- Existential quantifier

$$\exists f \triangleq \begin{cases} 1 & \text{if } (\exists a \in \mathbb{A}) f(a) = 1 \\ 0 & \text{if } (\forall a \in \mathbb{A}) f(a) = 0 \\ \bot & \text{otherwise} \end{cases}$$

is a continuous function $\exists : (\mathbb{A}_\bot \to_{fs} 2_\bot) \to 2_\bot$,

because $(\forall a \in \mathbb{A}) f(a) = 0$ iff $f(a_1) = 0 \land \cdots \land f(a_n) = 0$

where $\text{supp } f \uplus \{a\} = \{a_1 \ldots, a_n\}$ (finite).
Commercial break

Nominal Sets
Names and Symmetry in Computer Science

Nominal Scott Domains

[Turner & Winskel, CSL 2009] [Lösch & Pitts, POPL 2013]

- Posets in topos of nominal sets.
- Require limits/continuity only for directed sets whose elements have a common finite support (‘uniform-directed’ subsets).

Associated notion of compactness is more liberal than classical one – replace ‘finite’ by ‘orbit-finite’

e.g. compact functions are joins of orbit-finite consistent sets of step functions
Nominal Scott Domains

[Turner & Winskel, CSL 2009] [Lösch & Pitts, POPL 2013]

- Posets in topos of nominal sets.
- Require limits/continuity only for directed sets whose elements have a common finite support (‘uniform-directed’ subsets).
- Category of NSDs is cartesian closed and has fixpoint recursion for both morphisms and objects. But it also models locally scoped names (and more besides).
Nominal Scott Domains

[Turner & Winskel, CSL 2009] [Lösch & Pitts, POPL 2013]

- Posets in topos of nominal sets.
- Require limits/continuity only for directed sets whose elements have a common finite support (‘uniform-directed’ subsets).
- Category of NSDs is cartesian closed and has fixpoint recursion for both morphisms and objects.

But it also models **locally scoped names** (and more besides).

Using morphisms \(\_ \setminus \_ : A \perp \times D \to D\) satisfying

\[
\begin{align*}
\perp \setminus d &= \perp \\
\left(a \setminus b\right) \setminus d &= b \setminus a \setminus d \\
a \notin \text{supp}(a \setminus d) \\
a \notin \text{supp} \; d &\Rightarrow a \setminus d = d
\end{align*}
\]
Higher-order computable functions + locally scoped names

Plotkin’s PCF

Programming language for Computable Functions:
simply typed $\lambda$-calculus over ground types $\text{bool}$ & $\text{nat}$, with arithmetic and boolean operations and fixpoint recursion.

Higher-order computable functions
+ locally scoped names

\( \nu \text{PCF} = \text{Plotkin's PCF} \) extended with type \( \text{name} + \cdots \)

Denotational semantics using nominal Scott domains:

\[
\begin{align*}
\llbracket \text{name} \rrbracket & \triangleq \mathbb{A}_\bot \\
\llbracket \text{bool} \rrbracket & \triangleq \mathbb{2}_\bot \\
\llbracket \text{nat} \rrbracket & \triangleq \mathbb{N}_\bot \\
\llbracket \tau \rightarrow \tau' \rrbracket & \triangleq \llbracket \tau \rrbracket \rightarrow_{\text{fs}} \llbracket \tau' \rrbracket
\end{align*}
\]

supports the interpretation of terms for name-equality test, name-swapping and locally scoped names, \( \nu a.e \)
Higher-order computable functions + locally scoped names

\( \nu \text{PCF} = \) Plotkin’s \textbf{PCF} extended with type \textit{name} + \cdots

Denotational semantics using nominal Scott domains:

\[
\begin{align*}
\llbracket \text{name} \rrbracket & \triangleq A_{\perp} \\
\llbracket \text{bool} \rrbracket & \triangleq 2_{\perp} \\
\llbracket \text{nat} \rrbracket & \triangleq \mathbb{N}_{\perp} \\
\llbracket \tau \to \tau' \rrbracket & \triangleq \llbracket \tau \rrbracket \to_{fs} \llbracket \tau' \rrbracket
\end{align*}
\]

supports the interpretation of terms for name-equality test, name-swapping and locally scoped names, \( \nu a. e \), done \textit{Odersky-style}

characteristic feature is the conversion

\( \nu a. \lambda x : \tau. e = \lambda x : \tau. \nu a. e \)
Local scoping example

[suggested by Tzevelekos]

\[ F_1 \triangleq \lambda q : (\text{name} \to \text{bool}) \to \text{bool}. \nu a. q \, \text{eq}_a \]
\[ F_2 \triangleq \lambda q : (\text{name} \to \text{bool}) \to \text{bool}. \, q \, k_F \]

are contextually equivalent \( \nu \text{PCF} \) terms of type

\[ ((\text{name} \to \text{bool}) \to \text{bool}) \to \text{bool} \]

where

\[ \left\{ \begin{array}{l}
\text{eq}_a \triangleq \lambda x : \text{name}. \text{if } x = a \text{ then } T \text{ else } F \\
\text{k}_F \triangleq \lambda x : \text{name}. F
\end{array} \right. \]
Local scoping example

[suggested by Τzevelekos]

\[ F_1 \triangleq \lambda q : (\text{name} \rightarrow \text{bool}) \rightarrow \text{bool}. \nu a. q \; \text{eq}_a \]

\[ F_2 \triangleq \lambda q : (\text{name} \rightarrow \text{bool}) \rightarrow \text{bool}. q \; k_F \]

are contextually equivalent \( \nu PCF \) terms of type

\[ ((\text{name} \rightarrow \text{bool}) \rightarrow \text{bool}) \rightarrow \text{bool} \]

where

\[
\begin{cases}
\text{eq}_a \triangleq \lambda x : \text{name}. \text{if } x = a \text{ then } T \text{ else } F \\
k_F \triangleq \lambda x : \text{name}. F
\end{cases}
\]

but \([F_1] \neq [F_2]\), because

\[
\begin{cases}
[F_1] \text{ exists} = 1 \\
[F_2] \text{ exists} = 0
\end{cases}
\]
Full abstraction for $\nu$PCF$^+$

Plotkin’s classic full abstraction result for PCF + por:

contextual preorder (operational)     information order (denotational)
\[ e \leq_{\text{ctx}} e' : \tau \quad \iff \quad \llbracket e \rrbracket \sqsubseteq \llbracket e' \rrbracket \]
Full abstraction for $\nu{\text{PCF}}^+$

Plotkin’s classic full abstraction result for $\text{PCF} + \text{por}$:

contextual preorder (operational) $\quad$ information order (denotational)

$$e \leq_{\text{ctx}} e' : \tau \iff \llbracket e \rrbracket \sqsubseteq \llbracket e' \rrbracket$$

**Theorem.** [Lösch & Pitts, POPL 2013]
The NSD model is fully abstract for $\nu{\text{PCF}}^+ = \nu{\text{PCF}}$ extended with

- parallel or $\text{por} : \text{bool} \to \text{bool} \to \text{bool}$
- exists name $\text{exists} : (\text{name} \to \text{bool}) \to \text{bool}$
- definite name description $\text{the} : (\text{name} \to \text{bool}) \to \text{name}$

\[
\text{definite name description} = \text{the} \triangleq \begin{cases} 
\text{unique } a \text{ s.t. } f a = 1, \text{ if it exists} \\
\bot, \text{ otherwise}
\end{cases}
\]
Full abstraction for $\nu$PCF$^+$

Plotkin's classic full abstraction result for PCF + por:

contextual preorder (operational)  \[ e \leq_{\text{ctx}} e' : \tau \quad \iff \quad \llbracket e \rrbracket \sqsubseteq \llbracket e' \rrbracket \]

Theorem. [Lösch & Pitts, POPL 2013]
The NSD model is fully abstract for $\nu$PCF$^+ = \nu$PCF extended with

- parallel or \( \text{por}: \text{bool} \to \text{bool} \to \text{bool} \)
- exists name \( \text{exists}:(\text{name} \to \text{bool}) \to \text{bool} \)
- definite name description \( \text{the}:(\text{name} \to \text{bool}) \to \text{name} \)

Proof is tricky. (E.g. have to reduce certain permutation-indexed joins to use of exists-name.)
Full abstraction for $\nu\text{PCF}^+$

Plotkin’s classic **full abstraction** result for $\text{PCF} + \text{por}$:

- contextual preorder (operational)
  \[ e \leq_{\text{ctx}} e' : \tau \]
- information order (denotational)
  \[ [e] \sqsubseteq [e'] \]

**Theorem.** [Lösch & Pitts, POPL 2013]
The NSD model is fully abstract for $\nu\text{PCF}^+ = \nu\text{PCF}$ extended with

- parallel or
  \[ \text{por} : \text{bool} \to \text{bool} \to \text{bool} \]
- exists name
  \[ \text{exists} : (\text{name} \to \text{bool}) \to \text{bool} \]
- definite name description
  \[ \text{the} : (\text{name} \to \text{bool}) \to \text{name} \]

Result extends to a language with name abstraction types and nominal data types that use them – cf. FreshML.
We understand ‘orbit-finite’ category-theoretically (= finitely presentable), but don’t really understand its logical status: how to predict where to replace ‘finite’ by ‘orbit-finite’ in computation theory?

[N.B. we can make ‘finite support’ automatic by working in choice-free classical HOL/set theory.]

Permutations of $A$ (= name-inequality symmetry) is not the only group of interest – useful to consider automorphisms of various relational structures on $A$ (linear orders, undirected graphs, . . . ).

See recent work of Bojańczyk et al on automata theory over infinite alphabets.