Full Abstraction for Nominal Scott Domains

Steffen Lösch and Andrew M. Pitts

UNIVERSITY OF CAMBRIDGE

Computer Laboratory
Nominal Semantics
Nominal Semantics

classical semantic theory (PCF)
Nominal Semantics

classical semantic theory (PCF)

operational semantics
Nominal Semantics

- classical semantic theory (PCF)
  - operational semantics
  - denotational semantics
Nominal Semantics

- Classical semantic theory (PCF)
- Operational semantics
- Adequacy
- Denotational semantics

Steffen Lösch and Andrew M. Pitts
Full Abstraction for Nominal Scott Domains
Nominal Semantics

- Classical semantic theory (PCF)
- Adequacy
- Full abstraction
- Operational semantics
- Denotational semantics

Steffen Lösch and Andrew M. Pitts
Full Abstraction for Nominal Scott Domains
Nominal Semantics

nominal semantic theory (PNA)

operational semantics  adequacy  denotational semantics

full abstraction
Nominal Semantics

nominal semantic theory (PNA)

operational semantics

adequacy

full abstraction

denotational semantics

increased expressivity

Steffen Lösch and Andrew M. Pitts  Full Abstraction for Nominal Scott Domains
PCF and PNA

\[ \lambda a. (\alpha a) b \]

\[ A(\lambda (\alpha a). V a) (V b) \]

capture-avoiding substitution

\[ e \mapsto e[e'/a] \]

fix

\[ \lambda (f: \text{term}) \lambda y: \text{term} \]

\[ \text{case } y' \text{ of } V x \text{ if } x = a \text{ then } e' \text{ else } y \mid A x_1 x_2 \]

\[ A(f x_1)(f x_2) \]

\[ L(x_1) L(\alpha a. f(x@a)) \]

Steffen Lösch and Andrew M. Pitts

Full Abstraction for Nominal Scott Domains
<table>
<thead>
<tr>
<th>meta-level</th>
<th>PCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>object-level</td>
<td>numbers</td>
</tr>
</tbody>
</table>
### PCF and PNA

<table>
<thead>
<tr>
<th></th>
<th><strong>meta-level</strong></th>
<th><strong>PCF</strong></th>
<th><strong>PNA</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>object-level</strong></td>
<td><strong>numbers</strong></td>
<td><strong>numbers, syntax with binding</strong></td>
</tr>
</tbody>
</table>

*Steffen Lösch and Andrew M. Pitts: Full Abstraction for Nominal Scott Domains*
## PCF and PNA

<table>
<thead>
<tr>
<th>meta-level</th>
<th>PCF</th>
<th>PNA</th>
</tr>
</thead>
<tbody>
<tr>
<td>object-level</td>
<td>numbers</td>
<td>numbers, syntax with binding</td>
</tr>
</tbody>
</table>

Object-level representation of \((\lambda a \rightarrow a) b\)
### PCF and PNA

<table>
<thead>
<tr>
<th>meta-level</th>
<th>PCF</th>
<th>PNA</th>
</tr>
</thead>
<tbody>
<tr>
<td>object-level</td>
<td>numbers</td>
<td>numbers, syntax with binding</td>
</tr>
</tbody>
</table>

Object-level representation of \((\lambda a \to a) \, b\)

\[
A \left( L \left( \alpha a. V a \right) \right) (V b)
\]
PCF and PNA

<table>
<thead>
<tr>
<th></th>
<th>PCF</th>
<th>PNA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>meta-level</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>object-level</strong></td>
<td>numbers</td>
<td>numbers, syntax with binding</td>
</tr>
</tbody>
</table>

Object-level representation of \((\lambda a \to a) b\)

\[ A (L (\alpha a. V a))(V b) \]

capture-avoiding substitution \(e \mapsto e[e'/a]\)
PCF and PNA

<table>
<thead>
<tr>
<th>meta-level</th>
<th>PCF</th>
<th>PNA</th>
</tr>
</thead>
<tbody>
<tr>
<td>object-level</td>
<td>numbers</td>
<td>numbers, syntax with binding</td>
</tr>
</tbody>
</table>

Object-level representation of \((\lambda a \rightarrow a) \ b\)

\[
A(\text{L}(\alpha a. \text{V} a))(\text{V} b)
\]

capture-avoiding substitution \(e \mapsto e[e'/a]\)

\[
\text{fix}(\lambda(f : \text{term} \rightarrow \text{term}) \rightarrow \lambda y : \text{term} \rightarrow \\
\text{case } y' \text{ of } \\
\quad \text{V } x \rightarrow \text{if } x = a \text{ then } e' \text{ else } y \\
\mid A x_1 x_2 \rightarrow A(f x_1)(f x_2) \\
\mid L x \rightarrow L(\alpha a. f(x @ a)))
\]
Symmetry and Names

Deal with: data and algorithms that are infinite but become finite when quotiented by a suitable notion of symmetry

atomic names $a \in A$

finite permutations $\pi: A \sim = A$

countably infinite set $\pi \in \text{Perm}$

$\lambda a a = \alpha\lambda b b = \alpha\lambda c c = \alpha...$

existential quantification for names $\exists a.e$

permutation action $\pi \cdot d: \text{Perm} \times D \to D$

satisfying $\text{id} \cdot d = d$

$\pi \cdot (\pi \cdot d) = (\pi \circ \pi) \cdot d$

Steffen Lösch and Andrew M. Pitts

Full Abstraction for Nominal Scott Domains
Symmetry and Names

Deal with: data and algorithms that are infinite but become finite
when quotiented by a suitable notion of symmetry
Symmetry and Names

Deal with: data and algorithms that are infinite but become finite when quotiented by a suitable notion of symmetry

atomic names $a \in A$
Symmetry and Names

Deal with: data and algorithms that are infinite but become finite when quotiented by a suitable notion of symmetry

countably infinite set

atomic names $a \in \mathbb{A}$
Symmetry and Names

Deal with: data and algorithms that are infinite but become finite when quotiented by a suitable notion of symmetry

countably infinite set

atomic names $a \in \mathbb{A}$

finite permutations $\pi : \mathbb{A} \cong \mathbb{A}$
Symmetry and Names

Deal with: data and algorithms that are infinite but become finite when quotiented by a suitable notion of symmetry

countably infinite set
atomic names $a \in \mathcal{A}$

our notion of symmetry
finite permutations $\pi : \mathcal{A} \cong \mathcal{A}$
Symmetry and Names

Deal with: data and algorithms that *are infinite* but become *finite* when *quotiented* by a suitable notion of symmetry

- countably infinite set
- atomic names \( a \in \mathbb{A} \)
- finite permutations \( \pi : \mathbb{A} \cong \mathbb{A} \)
- our notion of symmetry
- \( \in \text{Perm} \)

Steffen Lösch and Andrew M. Pitts
Full Abstraction for Nominal Scott Domains
Deal with: data and algorithms that are infinite but become finite when quotiented by a suitable notion of symmetry

atomic names \( a \in \mathbb{A} \)

finite permutations \( \pi : \mathbb{A} \cong \mathbb{A} \) 

\[ \lambda a \to a =_{\alpha} \lambda b \to b =_{\alpha} \lambda c \to c =_{\alpha} \ldots \]
Symmetry and Names

Deal with: data and algorithms that are infinite but become finite when quotiented by a suitable notion of symmetry.

- atomic names $a \in \mathbb{A}$
- finite permutations $\pi : \mathbb{A} \cong \mathbb{A}$

existential quantification for names $\text{ex } a. e$

$\lambda a \Rightarrow a =_\alpha \lambda b \Rightarrow b =_\alpha \lambda c \Rightarrow c =_\alpha \ldots$

$\lambda a \Rightarrow a =_\alpha \lambda b \Rightarrow b =_\alpha \lambda c \Rightarrow c =_\alpha \ldots$

existential quantification for names $\text{ex } a. e$
Symmetry and Names

Deal with: data and algorithms that are infinite but become finite when quotiented by a suitable notion of symmetry

countably infinite set

atomic names \( a \in \mathbb{A} \)

finite permutations \( \pi : \mathbb{A} \cong \mathbb{A} \in \text{Perm} \)

our notion of symmetry

\[
\begin{align*}
\lambda a & \to a =_a \lambda b \to b =_a \lambda c \to c =_a \ldots \\
\text{existential quantification for names } & \text{ex } a. e
\end{align*}
\]

\[
\text{permutation action } \pi \cdot d : \text{Perm} \times D \to D
\]

satisfying
Symmetry and Names

Deal with: data and algorithms that are infinite but become finite when quotiented by a suitable notion of symmetry

atomic names \( a \in A \)

finite permutations \( \pi : A \cong A \)

\( \lambda a \to a =_\alpha \lambda b \to b =_\alpha \lambda c \to c =_\alpha \ldots \)

existential quantification for names \( \text{ex} a. e \)

permutation action \( \pi \cdot d : \text{Perm} \times D \to D \)

satisfying

\( \text{id} \cdot d = d \)

\( \pi \cdot (\pi' \cdot d) = (\pi \circ \pi') \cdot d \)
Nominal Sets and Name Abstraction

finite support: \( d \in D \) is fin. supp. by \( A \subseteq fA \) if \( \pi \cdot d = d \) holds for all permutations \( \pi \) that preserve the elements of \( A \).

nominal set: set that has a permutation action and whose elements are finitely supported.

Examples: \( A, N, \lambda \)-terms.

name abstractions:

\[ \langle a \rangle_d \text{ is the equivalence class of } (a, d) \text{ under the relation } (a_1, d_1) \approx (a_2, d_2) := (a_1 b) \cdot d_1 = (a_2 b) \cdot d_2 \text{ for some fresh } b. \]

\[ [A] D := (A \times D) / \approx \text{ is the name abstraction set of } D. \]

partial name concretion operation: \( \langle a \rangle_d \@ b = (a b) \cdot d \), well-defined only if \( b \) is fresh.

nominal partial order (npo): nominal set with a partial order \( \sqsubseteq \) satisfying \( d \sqsubseteq d' \Rightarrow \pi \cdot d \sqsubseteq \pi \cdot d' \). If \( D, E \) are npos, so are \( D \times E \) and \([A] D\).
Nominal Sets and Name Abstraction

- **finite support**: $d \in D$ is fin. supp. by $A \subseteq_f A$ if $\pi \cdot d = d$ holds for all permutations $\pi$ that preserve the elements of $A$
Nominal Sets and Name Abstraction

- **finite support**: $d \in D$ is fin. supp. by $A \subseteq_{f} A$ if $\pi \cdot d = d$ holds for all permutations $\pi$ that preserve the elements of $A$

- **nominal set**: set that has a permutation action and whose elements are finitely supported
Nominal Sets and Name Abstraction

- **finite support**: $d \in D$ is fin. supp. by $A \subseteq_f A$ if $\pi \cdot d = d$ holds for all permutations $\pi$ that preserve the elements of $A$

- **nominal set**: set that has a permutation action and whose elements are finitely supported

Examples: $\mathbb{A}$, $\mathbb{N}$, $\lambda$-terms
Nominal Sets and Name Abstraction

- **finite support**: \( d \in D \) is fin. supp. by \( A \subseteq_f \Delta \) if \( \pi \cdot d = d \) holds for all permutations \( \pi \) that preserve the elements of \( A \)

- **nominal set**: set that has a permutation action and whose elements are finitely supported
  
  Examples: \( A, \mathbb{N}, \lambda \)-terms

**name abstractions**
Nominal Sets and Name Abstraction

- **finite support**: $d \in D$ is fin. supp. by $A \subseteq_f A$ if $\pi \cdot d = d$ holds for all permutations $\pi$ that preserve the elements of $A$
- **nominal set**: set that has a permutation action and whose elements are finitely supported
Examples: $A$, $\mathbb{N}$, $\lambda$-terms

**name abstractions**

- $\langle a \rangle d$ is the equivalence class of $(a, d)$ under the relation $\approx$ for some fresh $b$
Nominal Sets and Name Abstraction

- **finite support**: \( d \in D \) is fin. supp. by \( A \subseteq_f A \) if \( \pi \cdot d = d \) holds for all permutations \( \pi \) that preserve the elements of \( A \)

- **nominal set**: set that has a permutation action and whose elements are finitely supported

Examples: \( \mathbb{A}, \mathbb{N}, \lambda \)-terms

**name abstractions**

- \( \langle a \rangle d \) is the equivalence class of \((a, d)\) under the relation \( (a_1, d_1) \simeq (a_2, d_2) := (a_1 \ b) \cdot d_1 = (a_2 \ b) \cdot d_2 \) for some fresh \( b \)

swapping permutation
Nominal Sets and Name Abstraction

- **finite support**: $d \in D$ is fin. supp. by $A \subseteq_f A$ if $\pi \cdot d = d$ holds for all permutations $\pi$ that preserve the elements of $A$

- **nominal set**: set that has a permutation action and whose elements are finitely supported

Examples: $\mathbb{A}$, $\mathbb{N}$, $\lambda$-terms

**name abstractions**

- $\langle a \rangle d$ is the equivalence class of $(a, d)$ under the relation $(a_1, d_1) \approx (a_2, d_2) := (a_1 b) \cdot d_1 = (a_2 b) \cdot d_2$ for some fresh $b$

$b$ not in the support of $d_1$ and $d_2$
Nominal Sets and Name Abstraction

- **finite support**: $d \in D$ is fin. supp. by $A \subseteq_f A$ if $\pi \cdot d = d$ holds for all permutations $\pi$ that preserve the elements of $A$

- **nominal set**: set that has a permutation action and whose elements are finitely supported

Examples: $\mathbb{A}$, $\mathbb{N}$, $\lambda$-terms

**name abstractions**

- $\langle a \rangle d$ is the equivalence class of $(a, d)$ under the relation $(a_1, d_1) \approx (a_2, d_2) := (a_1 b) \cdot d_1 = (a_2 b) \cdot d_2$ for some fresh $b$

- $[\mathbb{A}]D := (\mathbb{A} \times D)/\approx$ is the name abstraction set of $D$

- swapping permutation

- $b$ not in the support of $d_1$ and $d_2$
Nominal Sets and Name Abstraction

- **finite support**: $d \in D$ is fin. supp. by $A \subseteq_f A$ if $\pi \cdot d = d$ holds for all permutations $\pi$ that preserve the elements of $A$
- **nominal set**: set that has a permutation action and whose elements are finitely supported
  Examples: $\mathbb{A}$, $\mathbb{N}$, $\lambda$-terms

**name abstractions**

- $\langle a \rangle d$ is the equivalence class of $(a, d)$ under the relation $(a_1, d_1) \sim (a_2, d_2) := (a_1 \ b) \cdot d_1 = (a_2 \ b) \cdot d_2$ for some fresh $b$
- $[\mathbb{A}]D := (\mathbb{A} \times D)/\sim$ is the **name abstraction set** of $D$
- partial **name concretion** operation: $\langle a \rangle d @ b = (a \ b) \cdot d$, well-defined only if $b$ is fresh
Nominal Sets and Name Abstraction

- **finite support**: \( d \in D \) is fin. supp. by \( A \subseteq_f A \) if \( \pi \cdot d = d \) holds for all permutations \( \pi \) that preserve the elements of \( A \)

- **nominal set**: set that has a permutation action and whose elements are finitely supported
  
  Examples: \( \mathbb{A} \), \( \mathbb{N} \), \( \lambda \)-terms

- **name abstractions**
  
  - \( \langle a \rangle d \) is the equivalence class of \((a, d)\) under the relation 
  
    \[
    (a_1, d_1) \approx (a_2, d_2) \Leftrightarrow (a_1 \ b) \cdot d_1 = (a_2 \ b) \cdot d_2 \text{ for some fresh } b
    \]
  
  - \( \llbracket A \rrbracket D := (A \times D)/\approx \) is the **name abstraction set** of \( D \)
  
  - partial **name concretion** operation: \( \langle a \rangle d @ b = (a \ b) \cdot d \), well-defined only if \( b \) is fresh

- **nominal partial order (npo)**: nominal set with a partial order \( \sqsubseteq \) satisfying \( d \sqsubseteq d' \Rightarrow \pi \cdot d \sqsubseteq \pi \cdot d' \)
Nominal Sets and Name Abstraction

- **finite support**: \( d \in D \) is fin. supp. by \( A \subseteq_f A \) if \( \pi \cdot d = d \) holds for all permutations \( \pi \) that preserve the elements of \( A \)

- **nominal set**: set that has a permutation action and whose elements are finitely supported
  
  Examples: \( \mathbb{A}, \mathbb{N}, \lambda \)-terms

**name abstractions**

- \( \langle a \rangle d \) is the equivalence class of \((a, d)\) under the relation \((a_1, d_1) \approx (a_2, d_2) : = (a_1 b) \cdot d_1 = (a_2 b) \cdot d_2\) for some fresh \( b \)

- \( [A]D := (A \times D)/\approx \) is the **name abstraction set** of \( D \)

- partial **name concretion** operation: \( \langle a \rangle d @ b = (a b) \cdot d \), well-defined only if \( b \) is fresh

- **nominal partial order (npo)**: nominal set with a partial order \( \sqsubseteq \) satisfying \( d \sqsubseteq d' \Rightarrow \pi \cdot d \sqsubseteq \pi \cdot d' \)

If \( D, E \) are npos, so are \( D \times E \) and \([A]D\)
Domain Theory and Approximation

...
Theme: build a nominal domain theory based on npos
Domain Theory and Approximation

Theme: build a nominal domain theory based on npos

Classically: approximation by directed sets

\[
\langle a \rangle \sqcup S = \sqcup \{ \langle a \rangle d \mid d \in S \}
\]

does not hold for all finitely-supported directed sets

\[
\langle a \rangle \sqcup S = \sqcup \{ \langle a \rangle d \mid d \in S \}
\]

holds now

\[a \text{ npo has joins of uniformly-supported directed sets iff it has joins of finitely-supported chains}\]

(Turner and Winskel, CSL 2009)
Domain Theory and Approximation

Theme: build a nominal domain theory based on npos

Classically: approximation by directed sets

$$\langle a \rangle \uparrow S = \bigcup \{ \langle a \rangle d \mid d \in S \}$$

does not hold for all finitely-supported directed sets

$$\langle a \rangle \uparrow S = \bigcup \{ \langle a \rangle d \mid d \in S \}$$

holds now

\[ a \text{ npo has joins of uniformly-supported directed sets iff it has joins of finitely-supported chains} \] (Turner and Winskel, CSL 2009)

\[ A \subseteq fA \]

obvious try

second try

leads naturally to the notion of nominal Scott domain
Domain Theory and Approximation

Theme: build a nominal domain theory based on npos

Classically: approximation by directed sets

\[ \langle a \rangle \uplus S = \uplus \{ \langle a \rangle d \mid d \in S \} \]

does not hold for all finitely-supported directed sets

\[ \langle a \rangle \uplus S = \uplus \{ \langle a \rangle d \mid d \in S \} \]

holds now

\[ a \text{npo has joins of uniformly-supported directed sets iff} \]
\[ \text{it has joins of finitely-supported chains} \]

(Turner and Winskel, CSL 2009)
Domain Theory and Approximation

Theme: build a nominal domain theory based on npos

Classically: approximation by directed sets

\[
\langle a \rangle \uplus S = \uplus \{ \langle a \rangle d \mid d \in S \}
\]

does not hold for all finitely-supported directed sets

\[
\langle a \rangle \uplus S = \uplus \{ \langle a \rangle d \mid d \in S \}
\]

holds now

\( a \) npo has joins of uniformly-supported directed sets iff it has joins of finitely-supported chains (Turner and Winskel, CSL 2009)
Domain Theory and Approximation

Theme: build a nominal domain theory based on npos

Classically: approximation by directed sets

\[ \langle a \rangle \sqcup S = \sqcup \{ \langle a \rangle d \mid d \in S \} \]

does not hold for all finitely-supported directed sets

uniformly-supported directed sets

\[ \langle a \rangle \sqcup S = \sqcup \{ \langle a \rangle d \mid d \in S \} \]

holds now

\[ a \text{npo has joins of uniformly-supported directed sets iff it has joins of finitely-supported chains} \]

(Turner and Winskel, CSL 2009)

\[ A \subseteq fA \]

obvious try

second try

\[ \Rightarrow \]

leads naturally to the notion of nominal Scott domain

Steffen Lösch and Andrew M. Pitts

Full Abstraction for Nominal Scott Domains
Domain Theory and Approximation

Theme: build a **nominal domain theory** based on npos

Classically: **approximation** by directed sets

- **finitely-supported directed sets**
  - $\langle a \rangle \sqcup S = \bigsqcup \{ \langle a \rangle d \mid d \in S \}$ does **not** hold for all finitely-supported directed sets $S$
Domain Theory and Approximation

Theme: build a nominal domain theory based on npos

Classically: approximation by directed sets

\[
\langle a \rangle \bigsqcup S = \bigsqcup \{ \langle a \rangle d \mid d \in S \}
\]
does not hold for all finitely-supported directed sets \( S \)

\[
\langle a \rangle \bigsqcup S = \bigsqcup \{ \langle a \rangle d \mid d \in S \}
\]
holds now

\[a \text{ npo has joins of uniformly-supported directed sets iff it has joins of finitely-supported chains}\] (Turner and Winskel, CSL 2009)

\[\alpha \approx \beta \implies \{ \alpha \} \sqsubseteq F \{ \beta \}\]

obvious try

\[\text{finitely-supported directed sets}
\]

\[\langle a \rangle \bigsqcup S = \bigsqcup \{ \langle a \rangle d \mid d \in S \}\]
does not hold for all finitely-supported directed sets \( S \)

\[\langle a \rangle \bigsqcup S = \bigsqcup \{ \langle a \rangle d \mid d \in S \}\]

\[\alpha \approx \beta \implies \{ \alpha \} \sqsubseteq F \{ \beta \}\]

\[\text{obvious try}
\]

\[\text{second try}\]

\[\implies \text{leads naturally to the notion of nominal Scott domain}\]
Domain Theory and Approximation

Theme: build a **nominal domain theory** based on npos

Classically: *approximation by directed sets*

- **finitely-supported directed sets**
  - $\langle a \rangle \sqcup S = \bigsqcup \{ \langle a \rangle^d \mid d \in S \}$ does **not** hold for all finitely-supported directed sets $S$

- **uniformly-supported directed sets**
  - $\langle a \rangle \sqcup S = \bigsqcup \{ \langle a \rangle^d \mid d \in S \}$ **holds** now

- A npo has joins of uniformly-supported directed sets if and only if it has joins of finitely-supported chains

TURNER AND WINSKEL, CSL 2009

Steffen Lösch and Andrew M. Pitts

Full Abstraction for Nominal Scott Domains
Theme: build a nominal domain theory based on npos

Classically: approximation by directed sets

- finitely-supported directed sets
  - \( \langle a \rangle \sqcup S = \bigsqcup \{ \langle a \rangle d \mid d \in S \} \) does not hold for all finitely-supported directed sets \( S \)

- uniformly-supported directed sets
  - \( \langle a \rangle \sqcup S = \bigsqcup \{ \langle a \rangle d \mid d \in S \} \) holds now
  - a npo has joins of uniformly-supported directed sets iff it has joins of finitely-supported chains
Theme: build a nominal domain theory based on npos

Classically: approximation by directed sets

- For all finitely-supported directed sets $S$, $\bigvee \{ \langle a \rangle d \mid d \in S \}$ does not hold.

- For uniformly-supported directed sets, $\bigvee \{ \langle a \rangle d \mid d \in S \}$ holds now.

A npo has joins of uniformly-supported directed sets iff it has joins of finitely-supported chains.
**Domain Theory and Approximation**

Theme: build a nominal domain theory based on npos

Classically: approximation by directed sets

- **finitely-supported directed sets**
  - \( \langle a \rangle \sqcup S = \bigsqcup \{ \langle a \rangle d \mid d \in S \} \) does not hold for all finitely-supported directed sets \( S \)

- **uniformly-supported directed sets**
  - \( \langle a \rangle \sqcup S = \bigsqcup \{ \langle a \rangle d \mid d \in S \} \) holds now

- a npo has joins of uniformly-supported directed sets iff it has joins of finitely-supported chains

---

Steffen Lösch and Andrew M. Pitts

Full Abstraction for Nominal Scott Domains
Domain Theory and Approximation

Theme: build a nominal domain theory based on npos

Classically: approximation by directed sets

- finitely-supported directed sets
  - \( \langle a \rangle \bigsqcup S = \bigsqcup \{ \langle a \rangle d \mid d \in S \} \) does not hold for all finitely-supported directed sets \( S \)

uniformly-supported directed sets

- \( \langle a \rangle \bigsqcup S = \bigsqcup \{ \langle a \rangle d \mid d \in S \} \) holds now
- a npo has joins of uniformly-supported directed sets iff it has joins of finitely-supported chains

(Turner and Winskel, CSL 2009)
Domain Theory and Approximation

Theme: build a nominal domain theory based on npos

Classically: approximation by directed sets

- obviously try
- second try

finitely-supported directed sets
- \( \langle a \rangle \bigsqcup S = \bigsqcup \{ \langle a \rangle d \mid d \in S \} \) does not hold for all finitely-supported directed sets \( S \)

uniformly-supported directed sets
- \( \langle a \rangle \bigsqcup S = \bigsqcup \{ \langle a \rangle d \mid d \in S \} \) holds now
- a npo has joins of uniformly-supported directed sets iff it has joins of finitely-supported chains

(Turner and Winskel, CSL 2009)

\( \Rightarrow \) leads naturally to the notion of nominal Scott domain

Steffen Lösch and Andrew M. Pitts

Full Abstraction for Nominal Scott Domains
PNA

Types

\[ \tau ::= \text{bool} \mid \text{nat} \mid \tau \times \tau \mid \tau \tau \mid \text{name} \mid \text{term} \mid \delta \tau \]

Expressions

\[ e \in \text{Exp} ::= x \mid T \mid F \mid \ldots \mid \lambda x e \mid \text{fix } e \mid \text{as for PCF} \]

\[ a \mid \alpha a. e \mid e \circ e \mid \text{name, abstraction and concretion} \]

\[ V e \mid A e e \mid L e \mid \lambda \text{-terms} \]

\[ \text{case } e \text{ of } (V x e | A x x e | L x e) \]

\[ \lambda \text{-term case } (e \leftrightarrow e) e \mid e = e \mid \nu \alpha. e \]

swapping, equality and restriction

▶ 2 kinds of identifiers

Variables

Names

\[ x \in V \]

\[ a \in A \]

can be substituted

can be permuted

\[ A \cap V = \emptyset \]

countably infinite

can be bound

Steffen Lösch and Andrew M. Pitts

Full Abstraction for Nominal Scott Domains
PNA: Programming with Name Abstractions
PNA: Programming with Name Abstractions

- Types \( \tau ::= \text{bool} \mid \text{nat} \mid \tau \times \tau \mid \tau \rightarrow \tau \mid \text{name} \mid \text{term} \mid \delta \tau \)
PNA: Programming with Name Abstractions

Types \( \tau \ ::= \text{bool} \mid \text{nat} \mid \tau \times \tau \mid \tau \rightarrow \tau \mid \text{name} \mid \text{term} \mid \delta \tau \)

Expressions
\( e \in \text{Exp} ::= x \mid T \mid F \mid \ldots \mid \lambda x \to e \mid \text{fix } e \mid \text{as for PCF} \)
\( a \mid \alpha a. e \mid e @ e \mid \text{name, abstraction and concretion} \)
\( V e \mid A e e \mid L e \mid \lambda \text{-terms} \)
\( \text{case } e \text{ of } (V x \to e \mid A x x \to e \mid L x \to e) \mid \lambda \text{-term case} \)
\( (e \equiv e) e \mid e = e \mid \nu a. e \mid \text{swapping, equality and restriction} \)
PNA: Programming with Name Abstractions

- **Types** \( \tau ::= \text{bool} \mid \text{nat} \mid \tau \times \tau \mid \tau \rightarrow \tau \mid \text{name} \mid \text{term} \mid \delta \tau \)

- **Expressions**

  \[ e \in \text{Exp} ::= x \mid T \mid F \mid \ldots \mid \lambda x \rightarrow e \mid \text{fix } e \mid \text{as for PCF} \]

  \[ a \mid \alpha a. e \mid e @ e \mid \text{name, abstraction and concretion} \]

  \[ V e \mid A e e \mid L e \mid \lambda\text{-terms} \]

  \[ \text{case } e \text{ of } (V x \rightarrow e \mid A x x \rightarrow e \mid L x \rightarrow e) \mid \lambda\text{-term case} \]

  \[ (e \leftrightarrow e) e \mid e = e \mid \nu a. e \mid \text{swapping, equality and restriction} \]

- **2 kinds of identifiers**

<table>
<thead>
<tr>
<th>Variables</th>
<th>Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x \in \mathcal{V} )</td>
<td>( a \in \mathcal{A} )</td>
</tr>
<tr>
<td>can be substituted</td>
<td>can be permuted</td>
</tr>
<tr>
<td>( \mathcal{A} \cap \mathcal{V} = \emptyset )</td>
<td></td>
</tr>
<tr>
<td>countably infinite</td>
<td></td>
</tr>
<tr>
<td>can be bound</td>
<td></td>
</tr>
</tbody>
</table>
Typing and Examples

\[ \Gamma \vdash e : \tau \]

\[ a \in A \]

\[ \Gamma \vdash a : \text{name} \]

\[ a \in A \]

\[ \Gamma \vdash e : \tau \]

\[ \alpha \]

\[ e : \delta \tau \]

\[ \Gamma \vdash e_1 : \delta \tau \]

\[ \Gamma \vdash e_2 : \text{name} \]

\[ \Gamma \vdash e_1 \mathbin{@} e_2 : \tau \]

\[ \Gamma \vdash e_1 \mathbin{\text{fix}} e_2 : \text{name} \]

\[ \Gamma \vdash V e : \text{term} \]

\[ \Gamma \vdash e_1 : \text{term} \]

\[ \Gamma \vdash e_2 : \text{term} \]

\[ \Gamma \vdash A e_1 e_2 : \text{term} \]

\[ \Gamma \vdash e : \delta \text{term} \]

\[ \Gamma \vdash L e : \text{term} \]

Object-level representation of \((\lambda a \ a)\ b\)

Capture-avoiding substitution \(e \mapsto e[e'/a]\)

\[ \text{fix} (\lambda \mathbin{\text{f}} : \text{term}\mathbin{\text{term}}) \lambda y : \text{term} \]

\[ \text{case } y \mathbin{\text{of}} V x \mathbin{\text{if }} x = a \text{then } e' \text{else } y | A x_1 x_2 \]

\[ A (f x_1) (f x_2) | L x \]

\[ L (\alpha a . f (x @ a)) \]

\[ \Gamma = \{ x_1 : \tau_1, \ldots, x_n : \tau_n \} \]
Typing and Examples

Typing judgement: $\Gamma \vdash e : \tau$
Typing and Examples

\[ \Gamma = \{ x_1 : \tau_1, \ldots, x_n : \tau_n \} \]

Typing judgement: \( \Gamma \vdash e : \tau \)
Typing and Examples

\[ \Gamma = \{ x_1 : \tau_1, \ldots, x_n : \tau_n \} \]

Typing judgement: \( \Gamma \vdash e : \tau \)

\[
\begin{align*}
  a & \in A \\
  \Gamma & \vdash a : \text{name}
\end{align*}
\]
Typing and Examples

Typing judgement: $\Gamma \vdash e : \tau$

$\Gamma = \{x_1 : \tau_1, \ldots, x_n : \tau_n\}$

$\frac{a \in A}{\Gamma \vdash a : \text{name}}$

$\frac{a \in A \quad \Gamma \vdash e : \tau}{\Gamma \vdash \alpha a. e : \delta \tau}$

Steffen Lösch and Andrew M. Pitts  
Full Abstraction for Nominal Scott Domains
Typing and Examples

Typing judgement: $\Gamma \vdash e : \tau$

- $a \in A \quad \Gamma \vdash a : \text{name}$
- $a \in A \quad \Gamma \vdash e : \tau \quad \Gamma \vdash \alpha a. e : \delta \tau$
- $\Gamma \vdash e_1 : \delta \tau \quad \Gamma \vdash e_2 : \text{name} \quad \Gamma \vdash e_1 \circ e_2 : \tau$

\[ \Gamma = \{ x_1 : \tau_1, \ldots, x_n : \tau_n \} \]
Typing and Examples

Typing judgement: $\Gamma \vdash e : \tau$

- $a \in A \quad \Gamma \vdash a : \text{name}$
- $\Gamma \vdash e : \tau \quad \Gamma \vdash \alpha a. e : \delta \tau$
- $\Gamma \vdash e_1 : \delta \tau \quad \Gamma \vdash e_2 : \text{name} \quad \Gamma \vdash e_1 @ e_2 : \tau$
- $\Gamma \vdash e : \text{name} \quad \Gamma \vdash e_1 : \text{term} \quad \Gamma \vdash e_2 : \text{term} \quad \Gamma \vdash e : \delta \text{term}$
- $\Gamma \vdash V e : \text{term} \quad \Gamma \vdash A e_1 e_2 : \text{term} \quad \Gamma \vdash L e : \text{term}$

$\Gamma = \{x_1 : \tau_1, \ldots, x_n : \tau_n\}$
Typing and Examples

Typing judgement: $\Gamma \vdash e : \tau$

- $a \in A \quad \Gamma \vdash a : \text{name}$
- $\Gamma \vdash e : \tau \quad \Gamma \vdash \alpha a. e : \delta \tau$
- $\Gamma \vdash e_1 : \delta \tau \quad \Gamma \vdash e_2 : \text{name}$
- $\Gamma \vdash e_1 @ e_2 : \tau$
- $\Gamma \vdash e : \text{name}$
- $\Gamma \vdash e_1 : \text{term} \quad \Gamma \vdash e_2 : \text{term}$
- $\Gamma \vdash e : \delta \text{term}$
- $\Gamma \vdash V e : \text{term}$
- $\Gamma \vdash A e_1 e_2 : \text{term}$
- $\Gamma \vdash L e : \text{term}$

Object-level representation of $(\lambda a \to a) b$

$A \left( L \left( \alpha a. V a \right) \right) (V b) : \text{term}$
Typing and Examples

Typing judgement: $\Gamma \vdash e : \tau$

<table>
<thead>
<tr>
<th>$a \in A$</th>
<th>$\Gamma \vdash a : \text{name}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Gamma \vdash e : \tau$</td>
<td>$\Gamma \vdash \alpha a.e : \delta \tau$</td>
</tr>
<tr>
<td>$\Gamma \vdash e_1 : \delta \tau$</td>
<td>$\Gamma \vdash e_2 : \text{name}$</td>
</tr>
<tr>
<td>$\Gamma \vdash e_1 @ e_2 : \tau$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\Gamma \vdash e : \text{name}$</th>
<th>$\Gamma \vdash e_1 : \text{term}$</th>
<th>$\Gamma \vdash e_2 : \text{term}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Gamma \vdash V e : \text{term}$</td>
<td>$\Gamma \vdash A e_1 e_2 : \text{term}$</td>
<td>$\Gamma \vdash L e : \text{term}$</td>
</tr>
</tbody>
</table>

Object-level representation of $(\lambda a \to a)\ b$

$A (L (\alpha a. V a)) (V b) : \text{term}$

capture-avoiding substitution $e \mapsto e[e'/a]$

$\text{fix}(\lambda(f: \text{term} \to \text{term}) \to \lambda y : \text{term} \to$

$\text{case}\ y'\ of$

$V x \to \text{if}\ x = a\ then\ e'\ else\ y$

$| A x_1 x_2 \to A(f x_1)(f x_2)$

$| L x \to L (\alpha a. f(x @ a)) : \text{term} \to \text{term}$
Denotational Semantics

Types denote nominal Scott domains:

\[ J \text{name} K = A \perp \]

\[ J \delta \tau K = [A] J \tau K \]

\[ J \tau_1 \tau_2 K = J \tau_1 K \cup, \text{fs} J \tau_2 K \]

Expressions denote uniform-continuous, finitely supported functions:

if \( \Gamma \vdash e : \tau \) holds, then 

\[ J e K \in J \Gamma K \cup, \text{fs} J \tau K \]

\[ J a K \rho = a \]

\[ J \alpha a e K \rho = \langle a \rangle (J e K \rho) \] if \( a \) is fresh for \( \rho \)

⇒ simple and straightforward definition in the nominal setting
Denotational Semantics

Types denote nominal Scott domains:

\[ J \text{name} K = A \bot \]

\[ J \delta \tau K = [A] J \tau K \]

\[ J \tau_1 \tau_2 K = J \tau_1 K \text{uc}, \text{fs} J \tau_2 K \]

Expressions denote uniform-continuous, finitely supported functions:

If \( \Gamma \vdash e : \tau \) holds, then \( J e K \in J \Gamma K \text{uc}, \text{fs} J \tau K \)

\[ J a K \rho = a \]

\[ \langle a \rangle (J e K \rho) \] if \( a \) is fresh for \( \rho \)

⇒ simple and straightforward definition in the nominal setting
Types denote nominal Scott domains:

1. \([\text{name}] = \mathbb{A}_\bot\)
Denotational Semantics

Types denote nominal Scott domains:

- $[\text{name}] = A_{\bot}$
- $[\delta \tau] = [A][\tau]$
Denotational Semantics

Types denote nominal Scott domains:

- $[\text{name}] = A \bot$
- $[\delta \tau] = \lbrack A \rbrack \lbrack \tau \rbrack$
- $[\tau_1 \to \tau_2] = [\tau_1] \to_{uc,fs} [\tau_2]$
Denotational Semantics

Types denote nominal Scott domains:

- $[\text{name}] = A_\bot$
- $[\delta \tau] = [A][\tau]$
- $[\tau_1 \rightarrow \tau_2] = [\tau_1] \rightarrow_{\text{uc,fs}} [\tau_2]$

Expressions denote uniform-continuous, finitely supported functions:
Denotational Semantics

Types denote nominal Scott domains:

1. \([\text{name}] = A \perp\)
2. \([\delta \tau] = [A][\tau]\)
3. \([\tau_1 \rightarrow \tau_2] = [\tau_1] \rightarrow_{\text{uc,fs}} [\tau_2]\)

Expressions denote uniform-continuous, finitely supported functions:

\[\text{if } \Gamma \vdash e : \tau \text{ holds, then } [e] \in [\Gamma] \rightarrow_{\text{uc,fs}} [\tau]\]
Types denote nominal Scott domains:
▷ $[\text{name}] = A \perp$
▷ $[\delta \tau] = [A][\tau]$
▷ $[\tau_1 \rightarrow \tau_2] = [\tau_1] \rightarrow_{\text{uc,fs}} [\tau_2]$

Expressions denote uniform-continuous, finitely supported functions:

if $\Gamma \vdash e : \tau$ holds, then $[e] \in [\Gamma] \rightarrow_{\text{uc,fs}} [\tau]$

▷ $[a] \rho = a$
Denotational Semantics

Types denote nominal Scott domains:

- $[\text{name}] = A_\bot$
- $[\delta \tau] = [A][\tau]$
- $[\tau_1 \rightarrow \tau_2] = [\tau_1] \rightarrow_{uc,fs} [\tau_2]$

Expressions denote uniform-continuous, finitely supported functions:

- if $\Gamma \vdash e : \tau$ holds, then $[e] \in [\Gamma] \rightarrow_{uc,fs} [\tau]$
- $[a]_\rho = a$
- $[\alpha a. e]_\rho = \langle a \rangle ([e]_\rho)$ if $a$ is fresh for $\rho$
Denotational Semantics

Types denote nominal Scott domains:

- $[\text{name}] = A_\bot$
- $[\delta \tau] = [A][\tau]$
- $[\tau_1 \to \tau_2] = [\tau_1] \to_{uc,fs} [\tau_2]$

Expressions denote uniform-continuous, finitely supported functions:

if $\Gamma \vdash e : \tau$ holds, then $[e] \in [\Gamma] \to_{uc,fs} [\tau]$

- $[[a]]\rho = a$
- $[[\alpha a.\ e]]\rho = \langle a \rangle ([e]\rho)$ if $a$ is fresh for $\rho$

$\Rightarrow$ simple and straight-forward definition in the nominal setting
Operational Semantics

Evaluation judgement:
\[ e \downarrow c \in A \]
\[ a \downarrow a \]
\[ e \downarrow c \]
\[ \alpha a \]
\[ e \downarrow \alpha a \]
\[ c e \downarrow c V \]
\[ e \downarrow V c \]
\[ e_1 \downarrow c_1 \]
\[ e_2 \downarrow c_2 \]
\[ A e_1 e_2 \downarrow A c_1 c_2 \]
\[ e \downarrow c L \]
\[ e \downarrow L c \]
\[ e \downarrow V c e_1 \left[ c / x_1 \right] \downarrow c' \]
\[ \text{case } e \text{ of } (V x_1 e_1 | \cdots) \downarrow c' \]

This semantics is adequate:
\[ J e_1 K = J e_2 K \Rightarrow e_1 \sim PNA e_2 \text{ iff } C \left[ e_1 \right] \downarrow c \Leftrightarrow C \left[ e_2 \right] \downarrow c \text{ for every ground context } C \left[ - \right] \]
Operational Semantics

Evaluation judgement: $e \Downarrow c$
Operational Semantics

Evaluation judgement: \( e \Downarrow c \)

variable-closed expression

This semantics is adequate:

\[ J e_1 K = J e_2 K \Rightarrow e_1 \sim PNA e_2 \iff C[ e_1 ] \Downarrow c \iff C[ e_2 ] \Downarrow c \]

for every ground context \( C[ - ] \).
Operational Semantics

Evaluation judgement: \( e \downarrow c \)

\[ c ::= \ldots \mid a \mid V\,c \mid A\,c\,c \mid L\,c \mid \alpha\,a\,c \]

variable-closed expression

\( \text{This semantics is adequate:} \)

\[ J[ e_1 ] = J[ e_2 ] \Rightarrow e_1 \sim = PNA e_2 \iff C[ e_1 ] \downarrow c \Leftrightarrow C[ e_2 ] \downarrow c \]
Operational Semantics

Evaluation judgement: $e \Downarrow c$

$c ::= \ldots | a | Vc | Ac\ c | L\ c | \alpha a.\ c$

variable-closed expression

$a \in \mathcal{A}$

$a \Downarrow a$
Operational Semantics

Evaluation judgement: $e \Downarrow c$

$c ::= \ldots | a | V \\ c | A c c | L c | \alpha a. c$

$a \in A$

\[
\frac{a \Downarrow a}{a \Downarrow a}
\]

\[
\frac{e \Downarrow c}{\alpha a. e \Downarrow \alpha a. c}
\]
Operational Semantics

**Evaluation judgement:**  \( e \Downarrow c \)

\[
\begin{align*}
  & a \in A \\
  & a \Downarrow a \\
  & e \Downarrow c \\
  & \alpha a. e \Downarrow \alpha a. c \\
  & e \Downarrow c \\
  & V e \Downarrow V c \\
  & e_1 \Downarrow c_1 \\
  & e_2 \Downarrow c_2 \\
  & A e_1 e_2 \Downarrow A c_1 c_2 \\
  & e \Downarrow c \\
  & L e \Downarrow L c
\end{align*}
\]
Operational Semantics

Evaluation judgement: \( e \downarrow c \)

\[
\begin{align*}
a \in A & \quad \frac{a \downarrow a}{\alpha a. e \downarrow \alpha a. c} \\
\frac{e \downarrow c}{\alpha a. e \downarrow \alpha a. c} & \quad \frac{V e \downarrow V c}{V e \downarrow V c} \\
\frac{e \downarrow c}{V e \downarrow V c} & \quad \frac{e_1 \downarrow c_1 \quad e_2 \downarrow c_2}{A e_1 e_2 \downarrow A c_1 c_2} \\
\frac{e \downarrow c}{L e \downarrow L c} & \quad \frac{e \downarrow V c \quad e_1[c/x_1] \downarrow c'}{\text{case } e \text{ of (} V x_1 \rightarrow e_1 \mid \cdots \text{) } \downarrow c'}
\end{align*}
\]

variable-closed expression

\[c ::= \ldots \mid a \mid V c \mid A c \mid L c \mid \alpha a. c\]
Operational Semantics

Evaluation judgement: $e \Downarrow c$

$c ::= \ldots | a | V c | A c c | L c | \alpha a. c$

\[
\frac{a \in A}{a \Downarrow a} \quad \frac{e \Downarrow c}{\alpha a. e \Downarrow \alpha a. c} \quad \frac{e \Downarrow c}{V e \Downarrow V c} \quad \frac{e_1 \Downarrow c_1 \quad e_2 \Downarrow c_2}{A e_1 e_2 \Downarrow A c_1 c_2} \\
\frac{e \Downarrow c}{L e \Downarrow L c} \quad \frac{e \Downarrow V c \quad e_1[c/x_1] \Downarrow c'}{\text{case } e \text{ of } (V x_1 \to e_1 \mid \cdots) \Downarrow c'}
\]

This semantics is **adequate**:

$\llbracket e_1 \rrbracket = \llbracket e_2 \rrbracket \Rightarrow e_1 \simeq_{\text{PNA}} e_2$
Operational Semantics

Evaluation judgement: \( e \Downarrow c \)

\[
\begin{align*}
a \in A & \quad \Rightarrow \quad a \Downarrow a \\
\text{\( e \Downarrow c \)} & \quad \Rightarrow \quad \alpha a. e \Downarrow \alpha a. c \\
\text{\( e \Downarrow c \)} & \quad \Rightarrow \quad V e \Downarrow V c \\
\text{\( e_1 \Downarrow c_1 \), \( e_2 \Downarrow c_2 \)} & \quad \Rightarrow \quad A e_1 e_2 \Downarrow A c_1 c_2 \\
\text{\( e \Downarrow c \)} & \quad \Rightarrow \quad L e \Downarrow L c \\
\text{\( e \Downarrow V c \), \( c'[c/x_1] \Downarrow c' \)} & \quad \Rightarrow \quad \text{case e of (V x_1 \rightarrow e_1 | \cdots) \Downarrow c'}
\end{align*}
\]

This semantics is **adequate**:

\[
[e_1] = [e_2] \Rightarrow e_1 \equiv_{\text{PNA}} e_2
\]

\( e_1 \equiv_{\text{PNA}} e_2 \) iff \( C[e_1] \Downarrow c \Leftrightarrow C[e_2] \Downarrow c \) for every ground context \( C[\_] \)
Full Abstraction

Full abstraction fails for PNA (as for PCF without por)

\[ J_1 \rho = J_2 \rho \iff e_1 \sim e_2 =_{PNA} \]

PNA+ = PNA + parallel-or + name-exists + definite-description

Existential quantification for names: \( \Gamma, x : \text{name} \vdash e : \text{bool} \)

\[ \exists x. e \]

\[
\begin{cases}
\text{true} & \text{if } \exists a \in A \ J e \rho[a/\rho] = \text{true} \\
\text{false} & \text{if } \forall a \in A \ J e \rho[a/\rho] = \text{false} \\
\bot & \text{otherwise}
\end{cases}
\]

\[ e[a/x] \Downarrow T \]

\[ e[a/x] \Downarrow F \]

PNA+ is fully abstract:

\[ J e_1 \rho = J e_2 \rho \iff e_1 \sim e_2 =_{PNA} \]

Steffen Lösch and Andrew M. Pitts  
Full Abstraction for Nominal Scott Domains
Full Abstraction

Full abstraction fails for PNA (as for PCF without por)

\[
[e_1] = [e_2] \not\equiv e_1 \equiv_{PNA} e_2
\]
Full Abstraction

**Full abstraction** fails for PNA (as for PCF without por)

\[[e_1] = [e_2] \not\iff e_1 \equiv_{\text{PNA}} e_2\]

\[\text{PNA}+ = \text{PNA} + \text{parallel-or} + \text{name-exists} + \text{definite-description}\]
Full Abstraction

**Full abstraction** fails for PNA (as for PCF without por)

\[ [e_1] = [e_2] \not\equiv e_1 \equiv_{\text{PNA}} e_2 \]

PNA+ = PNA + parallel-or + name-exists + definite-description

Existential quantification for names:

\[ \Gamma, x : \text{name} \vdash e : \text{bool} \]

\[ \Gamma \vdash \text{ex } x. \ e : \text{bool} \]
Full Abstraction

**Full abstraction** fails for PNA (as for PCF without \( \text{por} \))

\[
[e_1] = [e_2] \not\equiv e_1 \congPNA e_2
\]

PNA\(+ = \text{PNA} + \text{parallel-or} + \text{name-exists} + \text{definite-description} \)

Existential quantification for names:

\[
\Gamma, x : \text{name} \vdash e : \text{bool} \\
\Gamma \vdash \text{ex} \ x . \ e : \text{bool}
\]

\[
[\text{ex} \ x . \ e] \rho = \begin{cases} 
\text{true} & \text{if } (\exists a \in A) \ [e](\rho[x \mapsto a]) = \text{true} \\
\text{false} & \text{if } (\forall a \in A) \ [e](\rho[x \mapsto a]) = \text{false} \\
\bot & \text{otherwise}
\end{cases}
\]
Full Abstraction

**Full abstraction** fails for PNA (as for PCF without por)

\[ \lbrack e_1 \rbrack = \lbrack e_2 \rbrack \not\equiv e_1 \equiv_{PNA} e_2 \]

PNA+ = PNA + parallel-or + name-exists + definite-description

Existential quantification for names:

\[ \Gamma, x : \text{name} \vdash e : \text{bool} \]

\[ \Gamma \vdash \text{ex } x . \; e : \text{bool} \]

\[ \lbrack \text{ex } x . \; e \rbrack \rho = \begin{cases} 
  \text{true} & \text{if } (\exists a \in A) \; \lbrack e \rbrack(\rho[x \mapsto a]) = \text{true} \\
  \text{false} & \text{if } (\forall a \in A) \; \lbrack e \rbrack(\rho[x \mapsto a]) = \text{false} \\
  \bot & \text{otherwise} 
\end{cases} \]

\[ e[a/x] \downarrow T \]

\[ \text{ex } x . \; e \downarrow T \]

\[ a' \text{ fresh for } e \]

\[ (\forall b \in \text{supp}(e) \cup \{ a' \}) \; e[b/x] \downarrow F \]

\[ \text{ex } x . \; e \downarrow F \]
**Full Abstraction**

**Full abstraction** fails for PNA (as for PCF without por)

\[
[e_1] = [e_2] \not\equiv e_1 \cong_{\text{PNA}} e_2
\]

\[P\text{NA}+ = \text{PNA} + \text{parallel-or} + \text{name-exists} + \text{definite-description}\]

Existential quantification for names:

\[
\Gamma, x : \text{name} \vdash e : \text{bool}
\]

\[
\therefore \Gamma \vdash \text{ex } x. e : \text{bool}
\]

\[
[\text{ex } x. e]_{\rho} = \begin{cases} 
  \text{true} & \text{if } (\exists a \in A) \ [e](\rho[x \mapsto a]) = \text{true} \\
  \text{false} & \text{if } (\forall a \in A) \ [e](\rho[x \mapsto a]) = \text{false} \\
  \bot & \text{otherwise}
\end{cases}
\]

\[
e[a/x] \downarrow T \quad a' \text{ fresh for } e \quad (\forall b \in \text{supp}(e) \cup \{a'\}) \ e[b/x] \downarrow F
\]

\[
\text{ex } x. e \downarrow T \quad \text{ex } x. e \downarrow F
\]

PNA+ is fully abstract:

\[
[e_1] = [e_2] \iff e_1 \cong_{\text{PNA}+} e_2
\]
Orbit-Finiteness

Changing the notion of approximation also changes the form of compact (or finite, isolated) elements.

Theorems:
The compact elements of $\mathcal{P}X$ are exactly the finite subsets of $X$.
The uniform-compact elements of $\mathcal{P}_{fs}X$ are exactly the orbit-finite subsets of $X$.

$\triangleright$ orbit: $x$ and $x'$ are in the same orbit iff $x' = \pi \cdot x$ for some $\pi$.

$\triangleright$ orbit-finite subset: a subset is called orbit-finite if it is contained in finitely many orbits.

So there is the following analogy:
FINITE DIRECTED SETS $\sim$ ORBIT-FINITE UNIFORMLY-SUPPORTED DIRECTED NOMINAL SETS.
Orbit-Finiteness

Changing the notion of approximation also changes the form of compact (or finite, isolated) elements
Orbit-Finiteness

*Changing the notion of approximation also changes the form of compact (or finite, isolated) elements*

Theorems:
Changing the notion of approximation also changes the form of compact (or finite, isolated) elements

Theorems:

The compact elements of $\mathcal{P}X$ are exactly the finite subsets of $X$
Orbit-Finiteness

Changing the notion of approximation also changes the form of compact (or finite, isolated) elements

Theorems:

- The **compact** elements of $\mathcal{P}X$ are exactly the **finite** subsets of $X$
- The **uniform-compact** elements of $\mathcal{P}_{fs}X$ are exactly the **orbit-finite** subsets of $X$. 
Orbit-Finiteness

*Changing the notion of approximation also changes the form of compact (or finite, isolated) elements*

**Theorems:**

- The **compact elements** of $P_X$ are exactly the **finite** subsets of $X$.
- The **uniform-compact elements** of $P_{fs}X$ are exactly the **orbit-finite** subsets of $X$.

- **orbit**: $x$ and $x'$ are in the same orbit iff $x' = \pi \cdot x$ for some $\pi$. 
Changing the notion of approximation also changes the form of compact (or finite, isolated) elements

Theorems:

- The compact elements of $\mathcal{P}X$ are exactly the finite subsets of $X$.
- The uniform-compact elements of $\mathcal{P}_{fs}X$ are exactly the orbit-finite subsets of $X$.

- **orbit**: $x$ and $x'$ are in the same orbit iff $x' = \pi \cdot x$ for some $\pi$.
- **orbit-finite subset**: a subset is called orbit-finite if it is contained in finitely many orbits.
Changing the notion of approximation also changes the form of compact (or finite, isolated) elements

Theorems:

- The **compact** elements of $PX$ are exactly the **finite** subsets of $X$

- The **uniform-compact** elements of $P_{fs}X$ are exactly the **orbit-finite** subsets of $X$.

**orbit**: $x$ and $x'$ are in the same orbit iff $x' = \pi \cdot x$ for some $\pi$

**orbit-finite subset**: a subset is called orbit-finite if it is contained in finitely many orbits

So there is the following analogy

finite directed sets $\sim$ orbit-finite uniformly-supported directed nominal sets
Conclusions

Open problems

- Failure of full abstraction for PNA+
- Are all compact elements of PNA-domains definable?
- Fully abstract game semantics for PNA
- Recursive domain equations and Scott information systems

Future direction: nominal semantics for concurrent calculi
Conclusions

nominal semantic theory (PNA)

operational semantics

adequacy

full abstraction

denotational semantics

Open problems

▶ failure of full abstraction for PNA+
▶ are all compact elements of PNA-domains definable?
▶ fully abstract game semantics for PNA
▶ recursive domain equations and Scott information systems

Future direction: nominal semantics for concurrent calculi
Conclusions

Open problems
Conclusions

nominal semantic theory (PNA)

operational semantics

adequacy

full abstraction

denotational semantics

Open problems

- failure of full abstraction for $\text{PNA} + \text{por}$
Open problems

- failure of full abstraction for PNA+por
- are all compact elements of PNA-domains definable?
Conclusions

Open problems

- failure of full abstraction for PNA$+$por
- are all compact elements of PNA-domains definable?
- fully abstract game semantics for PNA
Conclusions

nominal semantic theory (PNA)

operational semantics

adequacy

full abstraction
denotational semantics

Open problems

- failure of full abstraction for PNA+por
- are all compact elements of PNA-domains definable?
- fully abstract game semantics for PNA
- recursive domain equations and Scott information systems
Conclusions

nominal semantic theory (PNA)

Open problems

- failure of full abstraction for PNA+por
- are all compact elements of PNA-domains definable?
- fully abstract game semantics for PNA
- recursive domain equations and Scott information systems

Future direction: nominal semantics for concurrent calculi
Backup Slides
Full Typing Rules – PCF Part

\[
\begin{align*}
(x : \tau) & \in \Gamma \\
\Gamma \vdash x : \tau
\end{align*}
\]
\[
\begin{align*}
c & = T \mid F \\
\Gamma \vdash c : \text{bool}
\end{align*}
\]
\[
\begin{align*}
\Gamma \vdash e_1 : \text{bool} & \quad \Gamma \vdash e_2 : \tau & \quad \Gamma \vdash e_3 : \tau \\
\Gamma & \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 : \tau
\end{align*}
\]
\[
\begin{align*}
\Gamma \vdash 0 : \text{nat}
\end{align*}
\]
\[
\begin{align*}
\Gamma \vdash e : \text{nat} & \quad \Gamma \vdash e : \text{nat} \\
\Gamma & \vdash \text{S } e : \text{nat} & \quad \Gamma & \vdash \text{pred } e : \text{nat} & \quad \Gamma & \vdash \text{zero } e : \text{bool}
\end{align*}
\]
\[
\begin{align*}
\Gamma \vdash e_1 : \tau_1 & \quad \Gamma \vdash e_2 : \tau_2 \\
\Gamma & \vdash (e_1, e_2) : \tau_1 \times \tau_2
\end{align*}
\]
\[
\begin{align*}
\Gamma \vdash e : \tau_1 \times \tau_2 & \quad \Gamma \vdash \text{fst } e : \tau_1 \\
\Gamma & \vdash \text{snd } e : \tau_2
\end{align*}
\]
\[
\begin{align*}
\Gamma, x : \tau & \vdash e : \tau' \\
\Gamma & \vdash \lambda x : \tau \to e : \tau \to \tau'
\end{align*}
\]
\[
\begin{align*}
\Gamma \vdash e_1 : \tau' & \quad \Gamma \vdash e_2 : \tau \\
\Gamma & \vdash e_1 \ e_2 : \tau'
\end{align*}
\]
\[
\begin{align*}
\Gamma \vdash e : \tau \to \tau \\
\Gamma & \vdash \text{fix } e : \tau
\end{align*}
\]

Steffen L¨osch and Andrew M. Pitts

Full Abstraction for Nominal Scott Domains
Full Typing Rules – PNA Part

\[
\begin{align*}
\frac{a \in A}{\Gamma \vdash a : \text{name}} & \quad \frac{a \in A}{\Gamma \vdash \nu a. e : \tau} \\
\frac{\Gamma \vdash e_1 : \text{name} \quad \Gamma \vdash e_2 : \text{name}}{\Gamma \vdash (e_1 \equiv e_2) e_3 : \tau} & \quad \frac{\Gamma \vdash e_1 : \text{name} \quad \Gamma \vdash e_2 : \text{name}}{\Gamma \vdash e_1 = e_2 : \text{bool}} \\
\frac{\Gamma \vdash e : \text{name}}{\Gamma \vdash \text{V} e : \text{term}} & \quad \frac{\Gamma \vdash e_1 : \text{term} \quad \Gamma \vdash e_2 : \text{term}}{\Gamma \vdash \text{A} e_1 e_2 : \text{term}} & \quad \frac{\Gamma \vdash e : \delta \text{term}}{\Gamma \vdash \text{L} e : \text{term}} \\
\frac{\Gamma \vdash e : \text{term} \quad \Gamma, x_1 : \text{name} \vdash e_1 : \tau}{\Gamma, x_1 : \text{name} \vdash e : \tau} & \quad \frac{\Gamma, x_2 : \text{term}, x_2' : \text{term} \vdash e_2 : \tau \quad \Gamma, x_3 : \delta \text{term} \vdash e_3 : \tau}{\Gamma \vdash \text{case} e \text{ of } (\text{V} x_1 \to e_1 \mid \text{A} x_2 x_2' \to e_2 \mid \text{L} x_3 \to e_3) : \tau} \\
\frac{a \in A \quad \Gamma \vdash e : \tau}{\Gamma \vdash \alpha a. e : \delta \tau} & \quad \frac{\Gamma \vdash e_1 : \delta \tau \quad \Gamma \vdash e_2 : \text{name}}{\Gamma \vdash e_1 \oplus e_2 : \tau}
\end{align*}
\]
Full Operational Semantics – PCF Part

\[
c = T \mid F \mid 0 \mid (e_1, e_2) \mid \lambda x : \tau \to e
\]
\[
c \Downarrow c
\]
\[
e \Downarrow c
\]
\[
S e \Downarrow S c
\]
\[
e_1 \Downarrow T \quad e_2 \Downarrow c
\]
\[
\text{if } e_1 \text{ then } e_2 \text{ else } e_3 \Downarrow c
\]
\[
e_1 \Downarrow F \quad e_3 \Downarrow c
\]
\[
\text{if } e_1 \text{ then } e_2 \text{ else } e_3 \Downarrow c
\]
\[
e \Downarrow S c
\]
\[
\text{pred } e \Downarrow c
\]
\[
\text{zero } e \Downarrow T
\]
\[
\text{zero } e \Downarrow F
\]
\[
\text{fst } e \Downarrow c
\]
\[
e \Downarrow (e_1, e_2) \quad e_2 \Downarrow c
\]
\[
\text{snd } e \Downarrow c
\]
\[
e_1 \Downarrow \lambda x : \tau \to e \quad e[e_2/x] \Downarrow c
\]
\[
e_1 \quad e_2 \Downarrow c
\]
\[
e(\text{fix } e) \Downarrow c
\]
\[
\text{fix } e \Downarrow c
\]
Full Operational Semantics – PNA Part

\[
\frac{a \in A}{a \downarrow a} \quad \frac{e \downarrow c}{a \setminus c := c'} \quad \frac{e_1 \downarrow a_1 \quad e_2 \downarrow a_2 \quad e_3 \downarrow c}{(e_1 \Rightarrow e_2) \ e_3 \downarrow (a_1 \ a_2) \cdot c}
\]

\[
\frac{e_1 \downarrow a \quad e_2 \downarrow a}{e_1 = e_2 \downarrow T} \quad \frac{e_1 \downarrow a \quad e_2 \downarrow a' \quad a \neq a'}{e_1 = e_2 \downarrow F} \quad \frac{e \downarrow c}{\alpha a. \ e \downarrow \alpha a. \ c}
\]

\[
\frac{e_1 \downarrow \alpha a. \ c \quad e_2 \downarrow a' \quad a \neq a'}{e_1 @ e_2 \downarrow c'} \quad \frac{\nu a. (a a') \cdot c \downarrow c'}{e \downarrow c} \quad \frac{\nu a. \ e \downarrow c}{V \ e \downarrow V \ c}
\]

\[
\frac{e_1 \downarrow c_1 \quad e_2 \downarrow c_2}{A \ e_1 \ e_2 \downarrow A \ c_1 \ c_2} \quad \frac{e \downarrow c}{L \ e \downarrow L \ c} \quad \frac{e \downarrow V \ c \quad e_1[c/x_1] \downarrow c'}{\text{case } e \text{ of } (V \ x_1 \rightarrow \ e_1 \mid \cdots) \downarrow c'}
\]

\[
\frac{e \downarrow A \ c \ c'}{\text{case } e \text{ of } (\cdots \mid A \ x_2 \ x_2' \rightarrow e_2 \mid \cdots) \downarrow c''}
\]

\[
\frac{e \downarrow L \ c \quad e_3[c/x_3] \downarrow c'}{\text{case } e \text{ of } (\cdots \mid L \ x_3 \rightarrow e_3) \downarrow c'}
\]
Full Operational Semantics – Name Restriction

\[ a \setminus c := c' \]

\[
c = T \mid F \mid 0 \mid S c'
\]
\[
a \setminus c := c
\]
\[
a \setminus (e_1, e_2) := (\nu a. e_1, \nu a. e_2)
\]

\[
a \setminus \lambda x : \tau \to e := \lambda x : \tau \to \nu a. e
\]
\[
a \setminus a' := a'
\]
\[
a \setminus V c := V c'
\]

\[
a \setminus c_1 := c_1'
\]
\[
a \setminus c_2 := c_2'
\]
\[
a \setminus A c_1 c_2 := A c_1' c_2'
\]
\[
a \setminus L c := L c'
\]

\[
a \setminus c := c'
\]
\[
a \neq a'
\]
\[
a \setminus \alpha a'. c := \alpha a'. c'
\]

This operation is partial because \( a \setminus a := c \) holds for no \( c \).
Types denote domains:

- $[\text{bool}] = 2_{\bot}$, the flat domain on a discrete nominal set with two elements, $2 = \{\text{true}, \text{false}\}$.
- $[\text{nat}] = \mathbb{N}_{\bot}$, the flat domain on the discrete nominal set of natural numbers, $\mathbb{N} = \{0, 1, 2, \ldots\}$.
- $[\tau \times \tau'] = [\tau] \times [\tau']$, the product of nominal Scott domains.
- $[\tau \rightarrow \tau'] = [\tau] \rightarrow_c [\tau']$, the nominal Scott domain of finitely supported, uniform-continuous functions.
- $[\text{name}] = \mathbb{A}_{\bot}$, the flat domain on the nominal set of atomic names, $\mathbb{A} = \{a, b, c, \ldots\}$.
- $[\text{term}] = (\Lambda_{\alpha})_{\bot}$, the flat domain on the nominal set of $\alpha$-equivalence classes of $\lambda$-terms,

$$
\Lambda_{\alpha} \triangleq \{ t ::= a \mid \lambda a.t \mid t t \} / =_{\alpha} \quad (\text{where } a \in \mathbb{A}).
$$

- $[\delta \tau] = [\mathbb{A}][\tau]$, the domain of name abstractions of the nominal Scott domain $[\tau]$. 
Expressions denote continuous functions:

\[
[x] \rho = \rho x \\
[T] \rho = \text{true} \quad [F] \rho = \text{false}
\]

\[
[\text{if } e_1 \text{ then } e_2 \text{ else } e_3] \rho = \begin{cases} 
[e_2] \rho & \text{if } [e_1] \rho = \text{true} \\
[e_3] \rho & \text{if } [e_1] \rho = \text{false} \\
\bot & \text{otherwise}
\end{cases}
\]

\[
[0] \rho = 0
\]

\[
[S \ e] \rho = \begin{cases} 
n + 1 & \text{if } [e] \rho = n \in \mathbb{N} \\
\bot & \text{otherwise}
\end{cases}
\]

\[
[\text{pred } e] \rho = \begin{cases} 
n & \text{if } [e] \rho = n + 1 \in \mathbb{N} \\
\bot & \text{otherwise}
\end{cases}
\]

\[
[\text{zero } e] \rho = \begin{cases} 
\text{true} & \text{if } [e] \rho = 0 \in \mathbb{N} \\
\text{false} & \text{if } [e] \rho = n + 1 \in \mathbb{N} \\
\bot & \text{otherwise}
\end{cases}
\]
\[(e_1, e_2)\] \(\rho = ([e_1]\rho, [e_2]\rho)\)

\[\text{fst } e\] \(\rho = \pi_1([e]\rho)\) \hspace{1cm} \[\text{snd } e\] \(\rho = \pi_2([e]\rho)\)

\[\lambda x : \tau \to e\] \(\rho = \lambda d \in [\tau]. [e]\rho[x \mapsto d]\)

\([e_1 e_2]\rho = [e_1]\rho([e_2]\rho)\)

\[\text{fix } e\] \(\rho = \text{fix}([e]\rho)\)

\[a\] \(\rho = a\)

\[\nu a. e\] \(\rho = a \setminus ([e]\rho)\) if \(a \neq \rho\)

\([e_1 \equiv e_2] e_3\] \(\rho = \begin{cases} (a_1 a_2) \cdot ([e_3]\rho) & \text{if } [e_i]\rho = a_i \in A \ (i = 1, 2) \\ \bot & \text{otherwise} \end{cases}\)

\([e_1 = e_2]\rho = \begin{cases} eq_a([e_2]\rho) & \text{if } [e_1]\rho = a \in A \\ \bot & \text{otherwise} \end{cases}\)
Full Denotational Semantics – Expressions (3)

\[ [\alpha a. \, e] \rho = \langle a \rangle ([e] \rho) \quad \text{if} \ a \neq \rho \]

\[ [e_1 \otimes e_2] \rho = ([e_1] \rho) \otimes ([e_2] \rho) \]

\[ [V \, e] \rho = \begin{cases} [a]_\alpha & \text{if} \ [e] \rho = a \in A \\ \bot & \text{otherwise} \end{cases} \]

\[ [A \ e_1 \ e_2] \rho = \begin{cases} [t_1 \, t_2]_\alpha & \text{if} \ [e_i] \rho = [t_i]_\alpha \in \Lambda_\alpha \ (i = 1, 2) \\ \bot & \text{otherwise} \end{cases} \]

\[ [L \, e] \rho = \begin{cases} [\lambda a.t]_\alpha & \text{if} \ [e] \rho = \langle a \rangle [t]_\alpha \in [A]\Lambda_\alpha \\ \bot & \text{otherwise} \end{cases} \]

\[ [\text{case } e \text{ of } (V \, x_1 \rightarrow e_1 \mid A \, x_2 \, x'_2 \rightarrow e_2 \mid L \, x_3 \rightarrow e_3)] \rho = \]

\[ \begin{cases} [e_1] \rho[x_1 \rightarrow a] & \text{if} \ [e] \rho = [a]_\alpha \\ [e_2] \rho[x_2 \rightarrow [t]_\alpha, x'_2 \rightarrow [t']_\alpha] & \text{if} \ [e] \rho = [t \, t']_\alpha \\ [e_3] \rho[x_3 \rightarrow \langle a \rangle [t]_\alpha] & \text{if} \ [e] \rho = [\lambda a.t]_\alpha \\ \bot & \text{otherwise} \end{cases} \]