Concurrent Domain Theory with Nominal Sets

Steffen Lösch

University of Cambridge, Computer Laboratory
Higher-Order Process Language (HOPLA) can express (Higher-Order) CCS, Ambient Calculus with public names, denotational semantics, operational semantics, adequacy, and full abstraction. Cannot express: Local scoping.
Higher-Order Process Language (HOPLA)  
[Nygaard, Winskel]
Higher-Order Process Language (HOPLA)  
[Nygaard, Winskel]

can express

(Higher-Order) CCS  
Ambient Calculus with public names
**Introduction**

*Higher-Order Process Language (HOPLA)*

[Nygaard, Winskel]

*Can express:*

- (Higher-Order) CCS
- Ambient Calculus with public names

*Can express:

- Denotational semantics
- Operational semantics

*Adequacy:

- Full abstraction

Dagstuhl, October 2013  
Concurrent Domain Theory with Nominal Sets
Higher-Order Process LAnguage (HOPLA) [Nygaard, Winskel]

can express

(Higher-Order) CCS
Ambient Calculus with public names

adequacy

denotational semantics
operational semantics

full abstraction

Cannot express: *Local scoping*
Introduction (2)

Goal:
Goal:

Nominal HOPLA
[Turner, Winskel],[L., Winskel]
Goal:

Nominal HOPLA
[Turner, Winskel],[L., Winskel]

can express

$\pi$-calculus

full Ambient Calculus
Goal:

Nominal HOPLA
[Turner, Winskel],[L., Winskel]

can express

\(\pi\)-calculus\quad full Ambient Calculus

adequacy

nominal denotational semantics\quad operational semantics

full abstraction
HOPLA Domain Theory

Based on preorders

\[ \langle P, \preceq \rangle \]

Intuition: \( p_1, p_2 \in P \) are paths in a transition system and \( p_1 \preceq p_2 \) means that \( p_2 \) extends \( p_1 \).

Domains for HOPLA:

\[ \hat{P} \xrightarrow{\triangleright} \hat{P} := \{ X \downarrow | X \subseteq P \} \]

and \( X \downarrow := \{ p | (\exists p' \in X) p \preceq p' \} \)

\[ \triangleright \text{all joins exist in } \hat{P} : \text{complete lattice} \]

\[ \triangleright \text{preorder } ! P := \{ P | P \subseteq \text{finite } P \} \]

with \( P_1 \preceq ! P P_2 \iff P_1 \subseteq P_2 \downarrow \)

\[ \triangleright \text{P } \downarrow \text{ with } P \in ! P \text{ are the compact elements of } \hat{P} \]

Some categories (all have preorders \( P, Q \) as objects):

\[ \text{Pre} : \text{arrows are monotone functions } P \rightarrow Q \]

\[ \text{Lin} : \text{arrows are join-preserving functions } \hat{P} \rightarrow \hat{Q} \]

\[ \text{Cts} : \text{arrows are directed-join-preserving functions } \hat{P} \rightarrow \hat{Q} \]

Adjunctions:

\[ \text{Pre}(P, \hat{Q}) \sim = \text{Lin}(P, Q) \text{ and } \text{Lin}(! P, Q) \sim = \text{Cts}(P, Q) \]

Dagstuhl, October 2013

Concurrent Domain Theory with Nominal Sets
HOPLA Domain Theory

Based on preorders $\langle \mathbb{P}, \leq \rangle$
HOPLA Domain Theory

Based on preorders \( \langle \mathbb{P}, \leq_\mathbb{P} \rangle \)

**Intuition:** \( p_1, p_2 \in \mathbb{P} \) are paths in a transition system and \( p_1 \leq_\mathbb{P} p_2 \) means that \( p_2 \) extends \( p_1 \)
Based on preorders $\langle \mathbb{P}, \leq \rangle$

**Intuition:** $p_1, p_2 \in \mathbb{P}$ are paths in a transition system and $p_1 \leq_p p_2$ means that $p_2$ extends $p_1$

**Domains for HOPLA:** $\widehat{\mathbb{P}}$

- $\widehat{\mathbb{P}} := \{ X\downarrow \mid X \subseteq \mathbb{P} \}$ and $X\downarrow := \{ p \mid (\exists p' \in X) p \leq p' \}$
Based on preorders $\langle \mathbb{P}, \leq \mathbb{P} \rangle$

**Intuition:** $p_1, p_2 \in \mathbb{P}$ are paths in a transition system and $p_1 \leq_{\mathbb{P}} p_2$ means that $p_2$ extends $p_1$

**Domains for HOPLA:** $\widehat{\mathbb{P}}$

- $\widehat{\mathbb{P}} := \{ X \downarrow \mid X \subseteq \mathbb{P} \}$ and $X \downarrow := \{ p \mid (\exists p' \in X) p \leq p' \}$
- all joins exist in $\widehat{\mathbb{P}}$: complete lattice
Based on preorders $\langle P, \leq_P \rangle$

Intuition: $p_1, p_2 \in P$ are paths in a transition system and $p_1 \leq_P p_2$ means that $p_2$ extends $p_1$

Domains for HOPLA: $\hat{P}$

- $\hat{P} := \{ X\downarrow \mid X \subseteq P \}$ and $X\downarrow := \{ p \mid (\exists p' \in X) p \leq p' \}$
- all joins exist in $\hat{P}$: complete lattice
- preorder $!P := \{ P \mid P \subseteq_{\text{finite}} P \}$ with $P_1 \leq !_P P_2 \iff P_1 \subseteq P_2\downarrow$
HOPLA Domain Theory

Based on preorders $\langle \mathbb{P}, \leq_\mathbb{P} \rangle$

Intuition: $p_1, p_2 \in \mathbb{P}$ are paths in a transition system and $p_1 \leq_\mathbb{P} p_2$ means that $p_2$ extends $p_1$

Domains for HOPLA: $\hat{\mathbb{P}}$

- $\hat{\mathbb{P}} := \{ X \downarrow \mid X \subseteq \mathbb{P} \}$ and $X \downarrow := \{ p \mid (\exists p' \in X) p \leq p' \}$
- all joins exist in $\hat{\mathbb{P}}$: complete lattice
- preorder $\mathbb{!P} := \{ P \mid P \subseteq_{\text{finite}} \mathbb{P} \}$ with $P_1 \leq \mathbb{!P} P_2 \iff P_1 \subseteq P_2 \downarrow$
- $P \downarrow$ with $P \in \mathbb{!P}$ are the compact elements of $\hat{\mathbb{P}}$
HOPLA Domain Theory

Based on preorders \( \langle P, \leq_P \rangle \)

**Intuition:** \( p_1, p_2 \in P \) are paths in a transition system and \( p_1 \leq_P p_2 \) means that \( p_2 \) extends \( p_1 \)

Domains for HOPLA: \( \hat{P} \)

- \( \hat{P} := \{ X \downarrow \mid X \subseteq P \} \) and \( X \downarrow := \{ p \mid (\exists p' \in X) p \leq p' \} \)
- all joins exist in \( \hat{P} \): complete lattice
- preorder \( !P := \{ P \mid P \subseteq_{\text{finite}} P \} \) with \( P_1 \leq !P P_2 \iff P_1 \subseteq P_2 \downarrow \)
- \( P \downarrow \) with \( P \in !P \) are the compact elements of \( \hat{P} \)

Some categories (all have preorders \( P, Q \) as objects):

- \( \text{Pre} \): arrows are monotone functions \( P \rightarrow Q \)
- \( \text{Lin} \): arrows are join-preserving functions \( \hat{P} \rightarrow \hat{Q} \)
- \( \text{Cts} \): arrows are directed-join-preserving functions \( \hat{P} \rightarrow \hat{Q} \)

Adjunctions:
\( \text{Pre} (P, \hat{Q}) \sim = \text{Lin} (P, Q) \) and \( \text{Lin} (!P, Q) \sim = \text{Cts} (P, Q) \)

Dagstuhl, October 2013
Concurrent Domain Theory with Nominal Sets
**HOPLA Domain Theory**

Based on preorders $\langle P, \leq P \rangle$

**Intuition:** $p_1, p_2 \in P$ are paths in a transition system and $p_1 \leq P p_2$ means that $p_2$ extends $p_1$

**Domains for HOPLA:** $\widehat{P}$

- $\widehat{P} := \{ X \downarrow \mid X \subseteq P \}$ and $X \downarrow := \{ p \mid (\exists p' \in X) p \leq p' \}$
- all joins exist in $\widehat{P}$: complete lattice
- preorder $!P := \{ P \mid P \subseteq_{\text{finite}} P \}$ with $P_1 \leq !P P_2 \iff P_1 \subseteq P_2 \downarrow$
- $P \downarrow$ with $P \in !P$ are the compact elements of $\widehat{P}$

Some categories (all have preorders $P$, $Q$ as objects):

- **Pre:** arrows are monotone functions $P \rightarrow Q$
Intuition: $p_1, p_2 \in P$ are paths in a transition system and $p_1 \leq_P p_2$ means that $p_2$ extends $p_1$

Domains for HOPLA: $\hat{P}$

- $\hat{P} := \{X \downarrow \mid X \subseteq P\}$ and $X \downarrow := \{p \mid (\exists p' \in X) p \leq p'\}$
- all joins exist in $\hat{P}$: complete lattice
- preorder $!P := \{P \mid P \subseteq_{\text{finite}} P\}$ with $P_1 \leq !_P P_2 \iff P_1 \subseteq P_2 \downarrow$
- $P \downarrow$ with $P \in !_P$ are the compact elements of $\hat{P}$

Some categories (all have preorders $P, Q$ as objects):

- **Pre**: arrows are monotone functions $P \rightarrow Q$
- **Lin**: arrows are join-preserving functions $\hat{P} \rightarrow \hat{Q}$
HOPLA Domain Theory

Based on preorders \(<\mathbb{P}, \leq_{\mathbb{P}}\)

Intuition: \(p_1, p_2 \in \mathbb{P}\) are paths in a transition system and \(p_1 \leq_{\mathbb{P}} p_2\) means that \(p_2\) extends \(p_1\)

Domains for HOPLA: \(\hat{\mathbb{P}}\)

- \(\hat{\mathbb{P}} := \{X \downarrow \mid X \subseteq \mathbb{P}\}\) and \(X \downarrow := \{p \mid (\exists p' \in X) p \leq p'\}\)
- all joins exist in \(\hat{\mathbb{P}}\): complete lattice
- preorder \(!\mathbb{P} := \{P \mid P \subseteq_{\text{finite}} \mathbb{P}\}\) with \(P_1 \leq !\mathbb{P} P_2 \iff P_1 \subseteq P_2 \downarrow\)
- \(P \downarrow\) with \(P \in !\mathbb{P}\) are the compact elements of \(\hat{\mathbb{P}}\)

Some categories (all have preorders \(\mathbb{P}, \mathbb{Q}\) as objects):

- **Pre**: arrows are monotone functions \(\mathbb{P} \to \mathbb{Q}\)
- **Lin**: arrows are join-preserving functions \(\hat{\mathbb{P}} \to \hat{\mathbb{Q}}\)
- **Cts**: arrows are directed-join-preserving functions \(\hat{\mathbb{P}} \to \hat{\mathbb{Q}}\)
HOPLA Domain Theory

Based on preorders $\langle P, \leq_P \rangle$

Intuition: $p_1, p_2 \in P$ are paths in a transition system and $p_1 \leq_P p_2$ means that $p_2$ extends $p_1$

Domains for HOPLA: $\hat{P}$

- $\hat{P} := \{ X \downarrow \mid X \subseteq P \}$ and $X \downarrow := \{ p \mid (\exists p' \in X) \ p \leq p' \}$
- all joins exist in $\hat{P}$: complete lattice
- preorder $!P := \{ P \mid P \subseteq_{\text{finite}} P \}$ with $P_1 \leq !_P P_2 \iff P_1 \subseteq P_2 \downarrow$
- $P \downarrow$ with $P \in !_P$ are the compact elements of $\hat{P}$

Some categories (all have preorders $P, Q$ as objects):

- **Pre**: arrows are monotone functions $P \rightarrow Q$
- **Lin**: arrows are join-preserving functions $\hat{P} \rightarrow \hat{Q}$
- **Cts**: arrows are directed-join-preserving functions $\hat{P} \rightarrow \hat{Q}$

Adjunctions: $\text{Pre}(P, \hat{Q}) \simeq \text{Lin}(P, Q)$ and $\text{Lin}( !_P, Q) \simeq \text{Cts}(P, Q)$
The Structure of $\mathbf{Cts}$

- **Biproducts:** $P \sqcup Q$ is a biproduct in $\mathbf{Lin}$ and a product with injection arrows in $\mathbf{Cts}$.

- **Functions:** $\mathbf{Cts}$ is cartesian closed with function space $P \to Q := ! P^{\text{op}} \times Q$.

- **Fixed points:** the usual: $\text{fix}(f) = \bigcup_{n \in \mathbb{N}} f^n(\emptyset)$.

- **Bang:** The unit $\eta_P: \hat{P} \to \hat{! P}$ and counit $\epsilon_P: \hat{! P} \to \hat{P}$ of the adjunction $\mathbf{Lin}(!, Q) \sim = \mathbf{Cts}(P, Q)$ are

  $\eta_P X = \{ P \in ! P \mid P \subseteq X \}$

  $\epsilon_P X = \bigcup \{ P \downarrow \mid P \in X \}$

This gives us enough structure to model HOPLA.
The Structure of $\mathbf{Cts}$

- **Biproducts**: $P \oplus Q$ is a biproduct in $\mathbf{Lin}$ and a product with injection arrows in $\mathbf{Cts}$
The Structure of $\mathbf{Cts}$

- **Biproducts:** $\mathbb{P} \cup \mathbb{Q}$ is a biproduct in $\mathbf{Lin}$ and a product with injection arrows in $\mathbf{Cts}$

- **Functions:** $\mathbf{Cts}$ is cartesian closed with function space $\mathbb{P} \rightarrow \mathbb{Q} := \!\mathbb{P}^{\text{op}} \times \mathbb{Q}$
The Structure of $\mathbf{Cts}$

- **Biproducts:** $P \sqcup Q$ is a biproduct in $\mathbf{Lin}$ and a product with injection arrows in $\mathbf{Cts}$
- **Functions:** $\mathbf{Cts}$ is cartesian closed with function space $P \rightarrow Q := !P^{op} \times Q$
- **Fixed points:** the usual: $\text{fix}(f) = \bigcup_{n \in \mathbb{N}} f^n(\emptyset)$
The Structure of $\text{Cts}$

- **Biproducts:** $P \sqcup Q$ is a biproduct in $\text{Lin}$ and a product with injection arrows in $\text{Cts}$

- **Functions:** $\text{Cts}$ is cartesian closed with function space $P \to Q := !P \times Q$

- **Fixed points:** the usual: $\text{fix}(f) = \bigcup_{n \in \mathbb{N}} f^n(\emptyset)$

- **Bang:** The unit $\eta_P : \hat{P} \to \hat{!P}$ and counit $\epsilon_P : \hat{!P} \to \hat{P}$ of the adjunction $\text{Lin}(!P, Q) \cong \text{Cts}(P, Q)$ are

  $$\eta_P X = \{ P \in !P \mid P \subseteq X \}$$

  $$\epsilon_P X = \bigcup \{ P \downarrow \mid P \in X \}$$
The Structure of Cts

- **Biproducts:** $P \sqcup Q$ is a biproduct in $\text{Lin}$ and a product with injection arrows in $\text{Cts}$

- **Functions:** $\text{Cts}$ is cartesian closed with function space $P \to Q := !P^{\text{op}} \times Q$

- **Fixed points:** the usual: $\text{fix}(f) = \bigcup_{n \in \mathbb{N}} f^n(\emptyset)$

- **Bang:** The unit $\eta_P : \hat{P} \to \hat{!P}$ and counit $\epsilon_P : \hat{!P} \to \hat{P}$ of the adjunction $\text{Lin}( !P, Q ) \cong \text{Cts}(P, Q)$ are

  $$\eta_P X = \{ P \in !P \mid P \subseteq X \}$$

  $$\epsilon_P X = \bigcup \{ P \downarrow \mid P \in X \}$$

This gives us enough structure to model HOPLA.
Types:

\[ \tau ::= \tau | \tau \cup l \in L | \tau \cup l | T | \mu j \vec{T} \cup \tau | ! \tau | \bot \]

Expressions:

\[ e ::= x | \text{rec} x . e | \lambda x . e | e \cdot e | \Sigma i \in I e_i | l : e | \text{proj} l e | ! e | [e > c (x) => e] | \text{fold} e | \text{unfold} e | 0 \]

Actions:

\[ c ::= ! | e \mapsto \rightarrow c | l : c | \text{fold} c \]

Typing:

\[ \Gamma \vdash e : \tau \quad \text{and} \quad \tau : c : \tau' \]

Evaluation:

\[ e_c \rightarrow e' \]

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

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

\[ ! \tau : ! \tau : \tau' \]

\[ ! e \rightarrow e' \]

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

\[ \Gamma \vdash l' : e : \tau \cup l \in L \tau \cup l' : c : \tau' \]

\[ c' \rightarrow e' \]

\[ l' : e \mapsto \rightarrow \rightarrow c' \]

\[ ! e \rightarrow e' \]

\[ e_1 c \rightarrow e_3 e_2 \]

\[ \Gamma \vdash e_1 > c (x) => e_2 \]

\[ c' \rightarrow e' \]
Types:

\[ \tau ::= \tau \rightarrow \tau \mid \bigoplus_{l \in L} \tau_l \mid T \mid \mu j \vec{T}.\vec{\tau} \mid !\tau \mid \bot \]
### Types:
\[
\tau ::= \tau \to \tau \mid \bigoplus_{l \in L} \tau_l \mid T \mid \mu \overrightarrow{T} \cdot \overrightarrow{\tau} \mid !\tau \mid \bot
\]

### Expressions:
\[
e ::= x \mid \text{rec } x. \ e \mid \lambda x. \ e \mid e \ e \mid \Sigma_{i \in I} e_i \\
\mid l:e \mid \text{proj}_l \ e \mid !e \mid [e > c(x) => e] \\
\mid \text{fold } e \mid \text{unfold } e \mid 0
\]
**Types:** \[ \tau ::= \tau \rightarrow \tau \mid \bigoplus_{l \in L} \tau_l \mid T \mid \mu j \vec{T}.\vec{\tau} \mid !\tau \mid \perp \]

**Expressions:** \[ e ::= x \mid \text{rec } x. e \mid \lambda x. e \mid e \; e \mid \sum_{i \in I} e_i \mid l:e \mid \text{proj}_l e \mid !e \mid [e > c(x) => e] \mid \text{fold } e \mid \text{unfold } e \mid 0 \]

**Actions:** \[ c ::= ! \mid e \mapsto c \mid l:c \mid \text{fold } c \]
HOPLA: The Typed Process Calculus

\[ \text{Types:} \quad \tau ::= \tau \to \tau \mid \bigoplus_{l \in L} \tau_l \mid T \mid \mu \vec{T} \cdot \vec{\tau} \mid !\tau \mid \bot \]

\[ \text{Expressions:} \quad e ::= x \mid \text{rec } x. \ e \mid \lambda x. \ e \mid e \ e \mid \Sigma_{i \in I} e_i \]
\[ \mid l : e \mid \text{proj}_l \ e \mid !e \mid [e > c(x) \Rightarrow e] \]
\[ \mid \text{fold } e \mid \text{unfold } e \mid 0 \]

\[ \text{Actions:} \quad c ::= ! \mid e \mapsto c \mid l : c \mid \text{fold } c \]

\[ \text{Typing:} \quad \Gamma \vdash e : \tau \text{ and } \tau : c : \tau' \]
HOPLA: The Typed Process Calculus

<table>
<thead>
<tr>
<th>Types:</th>
<th>$\tau ::= \tau \rightarrow \tau \mid \bigoplus_{l \in L} \tau_l \mid T \mid \mu_j \vec{T} \cdot \vec{\tau} \mid !\tau \mid \bot$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expressions:</td>
<td>$e ::= x \mid \text{rec } x.\ e \mid \lambda x.\ e \mid e\ e \mid \Sigma_{i \in I} e_i \mid l:e \mid \text{proj}_l\ e \mid !e \mid [e &gt; c(x) =&gt; e] \mid \text{fold } e \mid \text{unfold } e \mid 0$</td>
</tr>
<tr>
<td>Actions:</td>
<td>$c ::= ! \mid e \mapsto c \mid l:c \mid \text{fold } c$</td>
</tr>
<tr>
<td>Typing:</td>
<td>$\Gamma \vdash e : \tau$ and $\tau : c : \tau'$</td>
</tr>
<tr>
<td>Evaluation:</td>
<td>$e \xrightarrow{c} e'$</td>
</tr>
</tbody>
</table>
Types: \[ \tau ::= \tau \rightarrow \tau \mid \bigoplus_{l \in L} \tau_l \mid T \mid \mu \vec{T}. \vec{\tau} \mid !\tau \mid \bot \]

Expressions: \[ e ::= x \mid \text{rec } x. e \mid \lambda x. e \mid e\ e \mid \Sigma_{i \in I} e_i \mid l : e \mid \text{proj}_l e \mid ![e] \mid [e > c(x) => e] \mid \text{fold } e \mid \text{unfold } e \mid 0 \]

Actions: \[ c ::= ![e] \mid e \mapsto c \mid l : c \mid \text{fold } c \]

Typing: \[ \Gamma \vdash e : \tau \text{ and } \tau : c : \tau' \]

Evaluation: \[ e \xrightarrow{c} e' \]
**HOPLA: The Typed Process Calculus**

**Types:**
\[
\tau ::= \tau \to \tau \mid \bigoplus_{l \in L} \tau_l \mid T \mid \mu \vec{j}. \vec{\tau} \mid !\tau \mid \bot
\]

**Expressions:**
\[
e ::= x \mid \text{rec } x. e \mid \lambda x. e \mid e e \mid \Sigma_{i \in I} e_i \mid l : e \mid \text{proj}_l e \mid !e \mid [e > c(x) \Rightarrow e] \mid \text{fold } e \mid \text{unfold } e \mid 0
\]

**Actions:**
\[
c ::= ! \mid e \mapsto c \mid l : c \mid \text{fold } c
\]

**Typing:**
\[
\Gamma \vdash e : \tau \text{ and } \tau : c : \tau'
\]

**Evaluation:**
\[
e \xrightarrow{c} e'
\]

\[\begin{align*}
\Gamma \vdash e : \tau & \quad \Gamma \vdash !e : !\tau \\
\Gamma \vdash e : \tau' & \quad \tau' : c : \tau' \quad l' \in L \\
\Gamma \vdash l': e : \bigoplus_{l \in L} \tau_l & \quad \bigoplus_{l \in L} \tau_l : (l': c) : \tau'
\end{align*}\]
HOPLA: The Typed Process Calculus

Types: $\tau ::= \tau \to \tau \mid \bigoplus_{l \in L} \tau_l \mid T \mid \mu_j \vec{T}. \vec{\tau} \mid ! \tau \mid \bot$

Expressions: $e ::= x \mid \text{rec } x. e \mid \lambda x. e \mid e \ e \mid \Sigma_{i \in I} e_i \mid l : e \mid \text{proj}_l e \mid ! e \mid [e > c(x) \Rightarrow e] \mid \text{fold } e \mid \text{unfold } e \mid 0$

Actions: $c ::= ! \mid e \mapsto c \mid l : c \mid \text{fold } c$

Typing: $\Gamma \vdash e : \tau$ and $\tau : c : \tau'$

Evaluation: $e \xrightarrow{c} e'$

\[
\frac{\Gamma \vdash e : \tau}{\Gamma \vdash ! e : ! \tau} \quad \frac{\Gamma \vdash e : \tau_{l'}}{\Gamma \vdash l' : e : \bigoplus_{l \in L} \tau_l} \quad \frac{! \tau : ! : \tau}{\frac{\tau_{l'} : c : \tau'}{l' \in L} \quad \frac{\bigoplus_{l \in L} \tau_l : (l' : c) : \tau'}{e \xrightarrow{c} e' \quad l' : e \xrightarrow{c} e'}}
\]

$e_1 \xrightarrow{c} e_3 \quad e_2[e_3/x] \xrightarrow{c'} e' \quad [e_1 > c(x) \Rightarrow e_2] \xrightarrow{c'} e'$
Example

CCS translation

▶ HOPLA type of a CCS process:

\[ \mu T \oplus l \in \{\tau\} \cup \{n | n \in \mathbb{N}\} \cup \{n | n \in \mathbb{N}\} =: S \]

Encoding of parallel composition:

\[
\begin{align*}
\text{rec} \ \lambda x. \lambda y. (\sum s \in S [x > s]:! (x') => s!:! (p x y)) + \\
\sum s \in S [y > s]:! (y') => s!:! (p x y')) + \\
\sum n \in \mathbb{N} [x > n]:! (x') => \tau!:! (p x y')) + \\
\sum n \in \mathbb{N} [y > n]:! (y') => \tau!:! (p x y'))
\end{align*}
\]
Example

CCS translation
Example

CCS translation

- HOPLA type of a CCS process:

\[
\mu T \cdot \bigoplus_{l \in \{\tau\} \cup \{n \mid n \in N\} \cup \{-n \mid n \in N\}} ! T =: S
\]
Example

CCS translation

- HOPLA type of a CCS process:

\[ \mu T. \bigoplus_{I \in \{\tau\} \cup \{n \mid n \in N\} \cup \{\bar{n} \mid n \in N\}} ! T \]

- Encoding of parallel composition:

\[
\begin{align*}
\text{rec } p \cdot \lambda x \cdot \lambda y \cdot (\sum_{s \in S} [x > s : ! (x') \Rightarrow s : ! (p x' y)]) \\
+ \sum_{s \in S} [y > s : ! (y') \Rightarrow s : ! (p x y')]
\end{align*}
\]

\[
+ \sum_{n \in \mathbb{N}} [x > n : ! (x') \Rightarrow [y > \bar{n} ! (y') : \Rightarrow \tau : ! (p x' y')]]
\]

\[
+ \sum_{n \in \mathbb{N}} [x > \bar{n} : ! (x') \Rightarrow [y > n : ! (y') \Rightarrow \tau : ! (p x' y')]]
\]
Full Abstraction for HOPLA

HOPLA may seem uncommon (or weird) at first sight
▶ 'concurrent' language but no parallel composition
▶ reason: language directly derived from its domain theory
▶ but: have full abstraction

Full abstraction:
\[ e_1 \sim \Theta e_2 \iff J e_1 K = J e_2 K \]

where \( e_1 \sim \Theta e_2 \) holds if for all contexts (expressions with a hole) \( C[\_] \) with \( \emptyset \vdash C[\_] : ! \perp \) we have
\[ C[e_1] ! \rightarrow e'_1 \iff C[e_2] ! \rightarrow e'_2 \]
Full Abstraction for HOPLA

HOPLA may seem uncommon (or weird) at first sight

- ‘concurrent’ language but no parallel composition
- reason: language directly derived from its domain theory
- but: have full abstraction
Full Abstraction for HOPLA

HOPLA may seem uncommon (or weird) at first sight
  ▶ ‘concurrent’ language but no parallel composition
  ▶ reason: language directly derived from its domain theory
  ▶ but: have full abstraction

Full abstraction:

\[ e_1 \simeq_{\text{ctx}} e_2 \iff \llbracket e_1 \rrbracket = \llbracket e_2 \rrbracket \]
HOPLA may seem uncommon (or weird) at first sight

- ‘concurrent’ language but no parallel composition
- reason: language directly derived from its domain theory
- but: have full abstraction

Full abstraction:

\[ e_1 \triangleq_{\text{ctx}} e_2 \iff [e_1] = [e_2] \]

where \( e_1 \triangleq_{\text{ctx}} e_2 \) holds if for all contexts (expressions with a hole) \( C[\_] \) with \( \emptyset \vdash C[e_i] : ! \perp \) we have

\[ C[e_1] \xrightarrow{!} e_1' \iff C[e_2] \xrightarrow{!} e_2' \]
Nominal HOPLA Domain Theory
Nominal HOPLA Domain Theory

Based on nominal preorders $\langle P, \leq_P \rangle$

$p_1 \leq_P p_2 \Rightarrow \pi \cdot p_1 \leq_P \pi \cdot p_2$
Nominal HOPLA Domain Theory

Based on nominal preorders $\langle \mathbb{P}, \leq_{\mathbb{P}} \rangle$

\[ p_1 \leq_{\mathbb{P}} p_2 \Rightarrow \pi \cdot p_1 \leq_{\mathbb{P}} \pi \cdot p_2 \]

Turner uses FM-preorders

Domains for nominal HOPLA:

\[ \hat{\mathbb{P}} \rightarrow \hat{\mathbb{P}} := \{ X \downarrow \mid X \subseteq \text{fs } \mathbb{P} \} \]

\[ X \downarrow := \{ p \mid (\exists p' \in X) \ p \leq p' \} \]

\[ \text{Turner uses FM-preorders} \]

Some categories (all have nominal preorders $\mathbb{P}$, $\mathbb{Q}$ as objects)

$\mathbb{N}_{\text{Pre}}$ (equivariant and monotone functions $\mathbb{P} \rightarrow \mathbb{Q}$)

$\mathbb{N}_{\text{Lin}}$ (equivariant and fs-join-preserving functions $\hat{\mathbb{P}} \rightarrow \hat{\mathbb{Q}}$)

$\mathbb{N}_{\text{Cts}}$ (equivariant and uniform-directed-join-preserving functions $\hat{\mathbb{P}} \rightarrow \hat{\mathbb{Q}}$)

$\mathbb{N}_{\text{Pre}}(\mathbb{P}, \hat{\mathbb{Q}}) \sim = \mathbb{N}_{\text{Lin}}(\mathbb{P}, \mathbb{Q})$ and $\mathbb{N}_{\text{Lin}}(\mathbb{P}, \mathbb{Q}) \sim = \mathbb{N}_{\text{Cts}}(\mathbb{P}, \mathbb{Q})$
Nominal HOPLA Domain Theory

Based on nominal preorders $\langle P, \leq_P \rangle$

$p_1 \leq_P p_2 \Rightarrow \pi \cdot p_1 \leq_P \pi \cdot p_2$

Domains for nominal HOPLA: $\hat{P}$

$\hat{P} := \{ X \downarrow | X \subseteq_{fs} P \}$ and $X \downarrow := \{ p | (\exists p' \in X) p \leq p' \}$

Turner uses FM-preorders

Some categories (all have nominal preorders $P, Q$ as objects)

$N_{Pre} :$ equivariant and monotone functions $P \rightarrow Q$

$N_{Lin} :$ equivariant and fs-join-preserving functions $\hat{P} \rightarrow \hat{Q}$

$N_{Cts} :$ equivariant and uniform-directed-join-preserving functions $\hat{P} \rightarrow \hat{Q}$

$N_{Pre}(P, \hat{Q}) \sim = N_{Lin}(P, Q)$ and $N_{Lin}(\hat{P}, Q) \sim = N_{Cts}(P, Q)$

Dagstuhl, October 2013  Concurrent Domain Theory with Nominal Sets
Nominal HOPLA Domain Theory

Based on nominal preorders $\langle P, \leq_P \rangle$

$p_1 \leq_P p_2 \Rightarrow \pi \cdot p_1 \leq_P \pi \cdot p_2$

Domains for nominal HOPLA: $\hat{P}$

- $\hat{P} := \{ X \downarrow \mid X \subseteq_{fs} P \}$ and $X \downarrow := \{ p \mid (\exists p' \in X) p \leq p' \}$
- all finitely supported joins exist in $\hat{P}$: complete lattice

Turner uses FM-preorders

Some categories (all have nominal preorders $P$, $Q$ as objects)

- $\mathbf{NPre} \colon$ equivariant and monotone functions $P \rightarrow Q$
- $\mathbf{NLin} \colon$ equivariant and fs-join-preserving functions $\hat{P} \rightarrow \hat{Q}$
- $\mathbf{NCts} \colon$ equivariant and uniform-directed-join-preserving functions $\hat{P} \rightarrow \hat{Q}$

$\mathbf{NPre}(P, \hat{Q}) \simeq = \mathbf{NLin}(P, Q)$ and $\mathbf{NLin}(\!\!P, Q) \simeq = \mathbf{NCts}(P, Q)$

Dagstuhl, October 2013

Concurrent Domain Theory with Nominal Sets
Nominal HOPLA Domain Theory

Based on nominal preorders \(⟨\mathbb{P}, \leq_\mathbb{P}⟩\)

\[ p_1 \leq_\mathbb{P} p_2 \Rightarrow \pi \cdot p_1 \leq_\mathbb{P} \pi \cdot p_2 \]

Domains for nominal HOPLA: \(\hat{\mathbb{P}}\)

- \(\hat{\mathbb{P}} := \{ X \downarrow \mid X \subseteq fs \mathbb{P} \}\) and \(X\downarrow := \{ p \mid (\exists p' \in X) p \leq p' \}\)
- all finitely supported joins exist in \(\hat{\mathbb{P}}\): complete lattice
- \(!\mathbb{P} := \{ P \mid P \subseteq_{\text{orbit-finite}} \mathbb{P} \}\) with \(P_1 \leq !_\mathbb{P} P_2 \iff P_1 \subseteq P_2\downarrow\)

Turner uses FM-preorders
Nominal HOPLA Domain Theory

Based on nominal preorders $\langle \mathbb{P}, \leq \rangle$

\[ p_1 \leq \mathbb{P} p_2 \Rightarrow \pi \cdot p_1 \leq \mathbb{P} \pi \cdot p_2 \]

Domains for nominal HOPLA: $\widehat{\mathbb{P}}$

- $\widehat{\mathbb{P}} := \{ X\downarrow \mid X \subseteq_{fs} \mathbb{P} \}$ and $X\downarrow := \{ p \mid (\exists p' \in X) \ p \leq p' \}$
- all finitely supported joins exist in $\widehat{\mathbb{P}}$: complete lattice
- $!\mathbb{P} := \{ P \mid P \subseteq_{orbit-finite} \mathbb{P} \}$ with $P_1 \leq !\mathbb{P} P_2 \iff P_1 \subseteq P_2\downarrow$
- $P\downarrow$ with $P \in !\mathbb{P}$ are the uniform-compact elements of $\widehat{\mathbb{P}}$

Turner uses FM-preorders
Nominal HOPLA Domain Theory

Based on nominal preorders \( \langle P, \leq_P \rangle \)

\[ p_1 \leq_P p_2 \Rightarrow \pi \cdot p_1 \leq_P \pi \cdot p_2 \]

Domains for nominal HOPLA: \( \hat{P} \)

- \( \hat{P} := \{ X \downarrow \mid X \subseteq_{fs} P \} \) and \( X\downarrow := \{ p \mid (\exists p' \in X) p \leq p' \} \)
- all finitely supported joins exist in \( \hat{P} \): complete lattice
- \( !P := \{ P \mid P \subseteq_{\text{orbit}-\text{finite}} P \} \) with \( P_1 \leq !P P_2 \iff P_1 \subseteq P_2\downarrow \)
- \( P\downarrow \) with \( P \in !P \) are the uniform-compact elements of \( \hat{P} \)

Some categories (all have nominal preorders \( P, Q \) as objects)

Turner uses FM-preorders

Dagstuhl, October 2013

Concurrent Domain Theory with Nominal Sets
Nominal HOPLA Domain Theory

Based on nominal preorders \( \langle P, \leq_P \rangle \)

\[ p_1 \leq_P p_2 \Rightarrow \pi \cdot p_1 \leq_P \pi \cdot p_2 \]

Domains for nominal HOPLA: \( \hat{P} \)

- \( \hat{P} := \{ X \downarrow \mid X \subseteq_{fs} P \} \) and \( X \downarrow := \{ p \mid (\exists p' \in X) p \leq p' \} \)
- all finitely supported joins exist in \( \hat{P} \): complete lattice
- \( !P := \{ P \mid P \subseteq_{\text{orbit-finite}} P \} \) with \( P_1 \leq !P P_2 \iff P_1 \subseteq P_2 \downarrow \)
- \( P \downarrow \) with \( P \in !P \) are the uniform-compact elements of \( \hat{P} \)

Some categories (all have nominal preorders \( P, Q \) as objects)

- **NPre**: equivariant and monotone functions \( P \rightarrow Q \)
Nominal HOPLA Domain Theory

Based on nominal preorders $\langle P, \leq_P \rangle$

\[ p_1 \leq_P p_2 \Rightarrow \pi \cdot p_1 \leq_P \pi \cdot p_2 \]

Domains for nominal HOPLA: $\hat{P}$

- $\hat{P} := \{ X \downarrow \mid X \subseteq_{fs} P \}$ and $X \downarrow := \{ p \mid (\exists p' \in X) p \leq p' \}$
- all finitely supported joins exist in $\hat{P}$: complete lattice
- $!P := \{ P \mid P \subseteq_{\text{orbit-finite}} P \}$ with $P_1 \leq !_P P_2 \iff P_1 \subseteq P_2 \downarrow$
- $P \downarrow$ with $P \in !_P$ are the uniform-compact elements of $\hat{P}$

Some categories (all have nominal preorders $P$, $Q$ as objects)

- **NPre**: equivariant and monotone functions $P \rightarrow Q$
- **NLin**: equivariant and $fs$-join-preserving functions $\hat{P} \rightarrow \hat{Q}$

Turner uses FM-preorders

Dagstuhl, October 2013

Concurrent Domain Theory with Nominal Sets
Nominal HOPLA Domain Theory

Based on nominal preorders \( \langle P, \leq_P \rangle \)

\[ p_1 \leq_P p_2 \Rightarrow \pi \cdot p_1 \leq_P \pi \cdot p_2 \]

Domains for nominal HOPLA: \( \hat{P} \)

1. \( \hat{P} := \{ X \downarrow \mid X \subseteq_{\text{fs}} P \} \) and \( X \downarrow := \{ p \mid (\exists p' \in X) p \leq p' \} \)
2. all finitely supported joins exist in \( \hat{P} \): complete lattice
3. \( !P := \{ P \mid P \subseteq_{\text{orbit-finite}} P \} \) with \( P_1 \leq !P P_2 \Leftrightarrow P_1 \subseteq P_2 \downarrow \)
4. \( P \downarrow \) with \( P \in !P \) are the uniform-compact elements of \( \hat{P} \)

Some categories (all have nominal preorders \( P, Q \) as objects)

- **NPre**: equivariant and monotone functions \( P \rightarrow Q \)
- **NLin**: equivariant and \( \text{fs}\)-join-preserving functions \( \hat{P} \rightarrow \hat{Q} \)
- **NCts**: equivariant and uniform-directed-join-preserving functions \( \hat{P} \rightarrow \hat{Q} \)

Turner uses FM-preorders

Dagstuhl, October 2013
Concurrent Domain Theory with Nominal Sets
Nominal HOPLA Domain Theory

Based on nominal preorders $\langle P, \leq_P \rangle$

$$p_1 \leq_P p_2 \Rightarrow \pi \cdot p_1 \leq_P \pi \cdot p_2$$

 Turner uses FM-preorders

Domains for nominal HOPLA: $\hat{P}$

$\hat{P} := \{ X \downarrow | X \subseteq_{fs} P \}$ and $X \downarrow := \{ p | (\exists p' \in X) p \leq p' \}$

- all finitely supported joins exist in $\hat{P}$: complete lattice
- $!P := \{ P | P \subseteq_{orbit-finite} P \}$ with $P_1 \leq !_P P_2 \iff P_1 \subseteq P_2 \downarrow$
- $P \downarrow$ with $P \in !_P$ are the uniform-compact elements of $\hat{P}$

Some categories (all have nominal preorders $P$, $Q$ as objects)

- $\text{NPre}$: equivariant and monotone functions $P \rightarrow Q$
- $\text{NLin}$: equivariant and fs-join-preserving functions $\hat{P} \rightarrow \hat{Q}$
- $\text{NCts}$: equivariant and uniform-directed-join-preserving functions $\hat{P} \rightarrow \hat{Q}$

$\text{NPre}(P, \hat{Q}) \cong \text{NLin}(P, Q)$ and $\text{NLin}(!P, Q) \cong \text{NCts}(P, Q)$
The Structure of NCts

▶ Biproducts, functions, fixed points and bang work just like before.

▶ Crucial isomorphism for name abstraction paths:

\[ A \hat{\sim} P = \hat{A} P \]

\[ \{ p \in A P | (N b) p \forall b \in (X \forall b) \} \]

\[ j X = fresh b \in \langle b \rangle \{ p \in P | \langle b \rangle p \in X \} \]

▶ Use this to model name abstraction and concretion in the language Jαa\[=\]i(⟨a⟩j)K

Je@a\[=\](jje)@a = \{p | ⟨a⟩p ∈ JeK} if a # jeK

Dagstuhl, October 2013
Concurrent Domain Theory with Nominal Sets
Biproducts, functions, fixed points and bang work just like before.

\[
\hat{A}^\hat{P} \cong \hat{A}^P_i \overset{\pi}{\sim} \{ p \in \hat{A}^P | (\text{N}b) \overset{\hat{b}}{\sim} \overset{\pi}{\text{X}} \} \\
\text{fresh} b \in \langle b \rangle \{ p \in P | \langle b \rangle \overset{\pi}{\text{P}} \in \text{X} \}
\]
The Structure of **NCts**

- Biproducts, functions, fixed points and bang work just like before.
- Crucial isomorphism for name abstraction paths:

\[
\begin{align*}
[A]\hat{P} & \cong [A]P \\
i & \cong \exists \\
j & \end{align*}
\]

- Use this to model name abstraction and concretion in the language $J\alpha a e K := i(\langle a \rangle J e K)$ and $J e @ a K := (j J e K) @ a = \{p | \langle a \rangle p \in J e K\}$ if $a \not\in J e K$.
Biproducts, functions, fixed points and bang work just like before.

Crucial isomorphism for name abstraction paths:

\[
\begin{align*}
[A] \hat{P} & \cong [A] \bar{P} \\
i & \quad \cong \\
j
\end{align*}
\]

\[
i X = \{ p \in [A] \bar{P} \mid (\forall b) \ p \odot b \in (X \odot b) \}
\]

\[
j X = \text{fresh } b \text{ in } \langle b \rangle \{ p \in \bar{P} \mid \langle b \rangle p \in X \}
\]
The Structure of NCts

- Biproducts, functions, fixed points and bang work just like before.
- Crucial isomorphism for name abstraction paths:

\[
[A]\hat{\mathcal{P}} \overset{i}{\underset{j}{\cong}} [A]\mathcal{P}
\]

\[
i X = \{ p \in [A]\mathcal{P} \mid (\forall b)\ p \odot b \in (X \odot b)\}
\]

\[
j X = \text{fresh } b \in \langle b \rangle\{ p \in \mathcal{P} \mid \langle b \rangle p \in X\}
\]

- Use this to model name abstraction and concretion in the language

\[
[\alpha a.\ e] := i(\langle a \rangle [e])
\]

\[
[e @ a] := (j [e]) @ a = \{ p \mid \langle a \rangle p \in [e]\} \quad \text{if } a \neq [e]
\]
Total Concretion in NCts

Partiality of concretion is a problem for the type-system for definability of all paths. Remedy: restriction operation on \( \hat{P} \):

\[
a \setminus \hat{X} := \{ p \in \hat{P} | (\langle N \rangle b) p \in (a b) \cdot X \}
\]

Examples:

\[
a \setminus \{a, b\} = \{b\}
\]

\[
a \setminus (A \setminus \{a, b\}) = A \setminus \{b\}
\]

Use this restriction operation to make concretion total:

\[
i (\langle b \rangle \setminus \hat{X}) \ @ \text{total}
\]

\[
a : = \{ \setminus X if a = b \}
\]

\[
b \setminus (a b) \cdot X \ otherwise
\]

This turns out to give \( \setminus X \ @ \text{total} = \{ p | \langle a \rangle p \in X \} \) (no side-condition).
Total Concretion in \textbf{NCts}

- Partiality of concretion is a problem
  - for the type-system
  - for definability of all paths

Examples:
- $a \{a, b\} = \{b\}$
- $a \setminus (A \setminus \{a, b\}) = A \setminus \{b\}$

Remedy: restriction operation on $\hat{P}$:

\[
a \setminus X := \{ p \in P \mid (\forall b \cdot p \in a \{b\} \cdot X) \}
\]

Use this restriction operation to make concretion total:

\[
\mathit{i}((\langle b \rangle X) @ \text{total}) a := \{ X \mid \langle a \rangle p \in X \}
\]

This turns out to give $X @ \text{total} a = \{ p \mid \langle a \rangle p \in X \}$ (no side-condition).
Partiality of concretion is a problem
  ▶ for the type-system
  ▶ for definability of all paths

Remedy: restriction operation on $\hat{P}$:

$$a \setminus X := \{ p \in P | (N_b) p \in (a \ast b) \cdot X \}$$

Examples:

$$a \setminus \{a, b\} = \{b\}$$

$$a \setminus (A - \{a, b\}) = A - \{b\}$$

Use this restriction operation to make concretion total:

$$i(\langle b \rangle X) @ total a := \{X \text{ if } a = b \}
\begin{array}{l}
  \text{otherwise}
\end{array}$$

This turns out to give

$$X @ total a = \{p | \langle a \rangle p \in X \}$$

(no side-condition)
Partiality of concretion is a problem
  ▶ for the type-system
  ▶ for definability of all paths
▶ Remedy: restriction operation on $\hat{P}$:

$$a\backslash X := \{ p \in P \mid (\forall b) p \in (a \cdot b) \cdot X \}$$

Examples:
- $a\{a, b\} = \{b\}$
- $a \backslash (A - \{a, b\}) = A - \{b\}$
Total Concretion in **NCts**

- Partiality of concretion is a problem
  - for the type-system
  - for definability of all paths
- **Remedy:** restriction operation on $\hat{P}$:

  $$a \backslash X := \{ p \in P \mid (\forall b) p \in (a \ b) \cdot X \}$$

Examples:

- $a \backslash \{a, b\} = \{b\}$
- $a \backslash (A - \{a, b\}) = A - \{b\}$
Partiality of concretion is a problem
  ▶ for the type-system
  ▶ for definability of all paths

**Remedy:** restriction operation on \( \widehat{P} \):

\[
a \setminus X := \{ p \in P \mid (\forall b) \ p \in (a \ b) \cdot X \}\]

Examples:
\[
\begin{align*}
  a \setminus \{a, b\} &= \{b\} \\
  a \setminus (\emptyset - \{a, b\}) &= \emptyset - \{b\}
\end{align*}
\]

Use this restriction operation to make concretion total:

\[
i(\langle b \rangle X) @^{\text{total}} a := \begin{cases} 
  X & \text{if } a = b \\
  b \setminus (a \ b) \cdot X & \text{otherwise}
\end{cases}
\]
Partiality of concretion is a problem
  - for the type-system
  - for definability of all paths

**Remedy:** restriction operation on \( \hat{P} \):

\[
a \setminus X := \{ p \in P \mid (\forall b) \ p \in (a \ b) \cdot X \}
\]

**Examples:**
- \( a \setminus \{a, b\} = \{b\} \)
- \( a \setminus (\mathbb{A} - \{a, b\}) = \mathbb{A} - \{b\} \)

Use this restriction operation to make concretion total:

\[
i(\langle b \rangle X) @^{\text{total}} a := \begin{cases} X & \text{if } a = b \\ b \setminus (\mathbb{A} - \{a, b\}) \cdot X & \text{otherwise} \end{cases}
\]

This turns out to give

\[
X @^{\text{total}} a = \{ p \mid \langle a \rangle p \in X \} \quad \text{(no side-condition)}
\]
Nominal HOPLA Syntax and Operational Semantics

Nominal Sets and Their Applications

- Nominal HOPLA Syntax and Operational Semantics
- Types: \( \tau \ ::= \ldots | \text{name } \otimes \tau | \delta \tau \)
- Expressions: \( e \ ::= \ldots | a \cdot e | \text{proj}_a e | \alpha.a.e | e@a | (\nu.a.e) | \ldots \)
- Actions: \( c \ ::= \ldots | a \cdot c | \alpha.a.c \)
- \( \Gamma \vdash e : \tau \)
- \( \Gamma \vdash \alpha.a.e : \delta \tau \)
- \( \tau : c : \tau' \)
- \( \delta \tau : \alpha.a.c : \delta \tau' \)
- \( e \cdot c \rightarrow e' \)
- \( \alpha.a.e \rightarrow \alpha.a.e' \)
- \( e@a \cdot c \rightarrow e@a' \)
- \( \nu.a.e \cdot c \rightarrow \nu.a.e' \)

Dagstuhl, October 2013
Concurrent Domain Theory with Nominal Sets
Nominal HOPLA Syntax and Operational Semantics

Types: \[ \tau ::= \ldots | \text{name} \otimes \tau | \delta \tau \]
Nominal HOPLA Syntax and Operational Semantics

Types:
\[ \tau ::= \ldots | \text{name} \otimes \tau | \delta \tau \]

Expressions:
\[ e ::= \ldots | a \cdot e | \text{proj}_a e | \alpha a. e | e @ a \\
| (\nu a. e) | \ldots \]
Nominal HOPLA Syntax and Operational Semantics

Types: \( \tau ::= \ldots \mid \text{name} \otimes \tau \mid \delta \tau \)

Expressions: \( e ::= \ldots \mid a \cdot e \mid \text{proj}_a e \mid \alpha a. e \mid e @ a \mid (\nu a. e) \mid \ldots \)

Actions: \( c ::= \ldots \mid a \cdot c \mid \alpha a. c \)
Nominal HOPLA Syntax and Operational Semantics

Types: \[ \tau ::= \ldots \mid \text{name} \otimes \tau \mid \delta \tau \]

Expressions: \[ e ::= \ldots \mid a \cdot e \mid \text{proj}_a e \mid \alpha a.e \mid e @ a \mid (\nu a.e) \mid \ldots \]

Actions: \[ c ::= \ldots \mid a \cdot c \mid \alpha a.c \]

\[ \begin{align*} 
\Gamma \vdash e : \tau & \quad \Rightarrow \quad \tau : c : \tau' \\
\Gamma \vdash \alpha a.e : \delta \tau & \quad \Rightarrow \quad \delta \tau : \alpha a.c : \delta \tau' \\
e \xrightarrow{c} e' & \quad \Rightarrow \quad \alpha a.e \xrightarrow{\alpha a.c} \alpha a.e' 
\end{align*} \]
Nominal HOPLA Syntax and Operational Semantics

Types: \( \tau ::= \ldots | \text{name} \otimes \tau | \delta \tau \)

Expressions: \( e ::= \ldots | a \cdot e | \text{proj} \_a e | \alpha a . e | e @ a | (\nu a . e) | \ldots \)

Actions: \( c ::= \ldots | a \cdot c | \alpha a . c \)

\[
\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \alpha a . e : \delta \tau}
\]
\[
\frac{\tau : c : \tau'}{\delta \tau : \alpha a . c : \delta \tau'}
\]
\[
\frac{e \xrightarrow{c} e'}{\alpha a . e \xrightarrow{\alpha a . c} \alpha a . e'}
\]
\[
\frac{e \xrightarrow{c} \alpha a . e'}{e @ a \xrightarrow{c} e'}
\]
Nominal HOPLA Syntax and Operational Semantics

Types: \[ \tau ::= \ldots \mid \text{name} \otimes \tau \mid \delta \tau \]

Expressions: \[ e ::= \ldots \mid a \cdot e \mid \text{proj}_a e \mid \alpha a. e \mid e \otimes a \mid (\nu a. e) \mid \ldots \]

Actions: \[ c ::= \ldots \mid a \cdot c \mid \alpha a. c \]

\[
\begin{array}{c}
\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \alpha a. e : \delta \tau}
\end{array}
\quad
\begin{array}{c}
\frac{\tau : c : \tau'}{\delta \tau : \alpha a. c : \delta \tau'}
\end{array}
\quad
\begin{array}{c}
\frac{e \xrightarrow{c} e'}{\alpha a. e \xrightarrow{\alpha a. c} \alpha a. e'}
\end{array}
\]
\[
\begin{array}{c}
\frac{\Gamma \vdash e : \delta \tau}{\Gamma \vdash e \otimes a : \tau}
\end{array}
\quad
\begin{array}{c}
\frac{}{-}
\end{array}
\quad
\begin{array}{c}
\frac{e \xrightarrow{c} e'}{e \otimes a \xrightarrow{c} e'}
\end{array}
\]
\[
\begin{array}{c}
\frac{\Gamma \vdash \nu a. e : \tau}{\Gamma \vdash \nu a. e : \tau}
\end{array}
\quad
\begin{array}{c}
\frac{}{-}
\end{array}
\quad
\begin{array}{c}
\frac{e \xrightarrow{c} e'}{\nu a. e \xrightarrow{c} \nu a. e'}
\end{array}
\]
\[
\begin{array}{c}
\frac{e \xrightarrow{c} e'}{\nu a. e \xrightarrow{c} \nu a. e'}
\end{array}
\quad
\begin{array}{c}
\frac{a \# c}{-}
\end{array}
\]

Dagstuhl, October 2013
Concurrent Domain Theory with Nominal Sets
Semantic Results for Nominal HOPLA

This gets us quite far. We have (without name restriction syntax)

\[ e_1 \sim = J_{\text{ctx}} K e_2 \Leftrightarrow J e_1 K \neq \emptyset \Leftrightarrow J e_2 K \neq \emptyset \]

\[ \Rightarrow \]

for adequacy \( J e_1 K = J e_2 K \Rightarrow e_1 \sim = \text{ctx} e_2 \)

Soundness of \( e_c \rightarrow e'_{\# c \nu a} \).

\[ \Rightarrow \]

\[ J ! \nu a . e K \subseteq J ! e K \]

but this does not hold!

Counter-example \( J e K = A - \{ a \} : J ! \nu a . e K = \eta (a \setminus (A - \{ a \})) = P_{\text{finite}} A \)

\[ J ! \nu a . e K = a \setminus (\eta (A - \{ a \})) = P_{\text{finite}} A \]
Semantic Results for Nominal HOPLA

This gets us quite far. We have (without name restriction syntax)
Semantic Results for Nominal HOPLA

This gets us quite far. We have (without name restriction syntax)
  ▶ soundness
Semantic Results for Nominal HOPLA

This gets us quite far. We have (without name restriction syntax)

▷ soundness

▷ a purely denotational full abstraction result:

\[ e_1 \equiv_{ctx} e_2 \iff [e_1] = [e_2] \]

where \( e_1 \equiv_{ctx} e_2 := (\forall C[-])[C[e_1]] \neq \emptyset \iff [C[e_2]] \neq \emptyset \)
This gets us quite far. We have (without name restriction syntax)

- soundness
- a purely denotational full abstraction result:

\[ e_1 \simeq_{[\text{ctx}]} e_2 \iff [e_1] = [e_2] \]

where \( e_1 \simeq_{[\text{ctx}]} e_2 := (\forall C[-]) \ [C[e_1]] \neq \emptyset \iff [C[e_2]] \neq \emptyset \)

\( \dashv \) for adequacy \(([e_1] = [e_2] \Rightarrow e_1 \simeq_{\text{ctx}} e_2)\) we seem to need name restriction in the syntax
This gets us quite far. We have (without name restriction syntax)

- soundness
- a purely denotational full abstraction result:

\[ e_1 \equiv_{ctx} e_2 \Leftrightarrow \llbracket e_1 \rrbracket = \llbracket e_2 \rrbracket \]

where \( e_1 \equiv_{ctx} e_2 := (\forall C[-]) \llbracket C[e_1] \rrbracket \neq \emptyset \Leftrightarrow \llbracket C[e_2] \rrbracket \neq \emptyset \)

for adequacy \((\llbracket e_1 \rrbracket = \llbracket e_2 \rrbracket \Rightarrow e_1 \equiv_{ctx} e_2\) we seem to need name restriction in the syntax

Soundness of \[ e \xrightarrow{c} e' \quad a \neq c \]

\[ \nu a. e \xrightarrow{c} \nu a. e' \]

requires \(\llbracket !\nu a. e \rrbracket \subseteq \llbracket \nu a. !e \rrbracket\)
This gets us quite far. We have (without name restriction syntax)

- soundness
- a purely denotational full abstraction result:
  \[ e_1 \simeq_{\text{ctx}} e_2 \iff \llbracket e_1 \rrbracket = \llbracket e_2 \rrbracket \]

where \( e_1 \simeq_{\text{ctx}} e_2 \) := \((\forall C[-]) \left[ C[e_1] \right] \neq \emptyset \iff \left[ C[e_2] \right] \neq \emptyset \)

\[ \vdash \text{for adequacy } \left( \llbracket e_1 \rrbracket = \llbracket e_2 \rrbracket \Rightarrow e_1 \simeq_{\text{ctx}} e_2 \right) \text{ we seem to need } \]

name restriction in the syntax

Soundness of \[ e \overset{c}{\rightarrow} e' \]

\[ \nu a. e \overset{c}{\rightarrow} \nu a. e' \]

requires \( \llbracket !\nu a. e \rrbracket \subseteq \llbracket \nu a. !e \rrbracket \)

but this does not hold!
This gets us quite far. We have (without name restriction syntax)

- soundness
- a purely denotational full abstraction result:

\[ e_1 \equiv_{\text{ctx}} e_2 \iff \llbracket e_1 \rrbracket = \llbracket e_2 \rrbracket \]

where \( e_1 \equiv_{\text{ctx}} e_2 := (\forall C[-]) [C[e_1]] \neq \emptyset \iff [C[e_2]] \neq \emptyset \)

‡ for adequacy (\( [e_1] = [e_2] \Rightarrow e_1 \equiv_{\text{ctx}} e_2 \)) we seem to need name restriction in the syntax

\[
\begin{align*}
\text{Soundness of} \quad & e \xrightarrow{c} e' \quad a \neq c \\
\nu a. \ e \xrightarrow{c} \nu a. \ e' \\
& \text{requires} \quad [!\nu a. \ e] \subseteq [\nu a. !e]
\end{align*}
\]

but this does not hold!

Counter-example \( [e] = A - \{a\} \):

\[
\begin{align*}
[!\nu a. \ e] &= \eta (a \setminus (A - \{a\})) = P_{fsA} \\
[\nu a. !e] &= a \setminus (\eta (A - \{a\})) = P_{\text{finiteA}}
\end{align*}
\]
Conclusions

Have: fully abstract domain theory for HOPLA

Want: nominal version for increased expressivity

Extension of domain theory quite straightforward

Promising partial results

Some problems with name restriction remain
Conclusions

- Have: fully abstract domain theory for HOPLA
Conclusions

- Have: fully abstract domain theory for HOPLA
- Want: nominal version for increased expressivity
Conclusions

- Have: fully abstract domain theory for HOPLA
- Want: nominal version for increased expressivity
- Extension of domain theory quite straight-forward
Conclusions

- Have: fully abstract domain theory for HOPLA
- Want: nominal version for increased expressivity
- Extension of domain theory quite straight-forward
- Promising partial results
Conclusions

- Have: fully abstract domain theory for HOPLA
- Want: nominal version for increased expressivity
- Extension of domain theory quite straight-forward
- Promising partial results
- Some problems with name restriction remain