Nominal Semantics of Abstraction and Restriction

Andrew Pitts
University of Cambridge
Computer Laboratory

Unofficial title:

The Joy of Name-Swapping

Andrew Pitts
University of Cambridge
Computer Laboratory

Aim: describe the topos of nominal sets
(a simple reformulation of Schanuel's atomic topos)

as a model for computations on syntactical structures involving

- freshness of names
  (as in "choose a fresh name")
- name-abstraction
  (e.g. as in λ-calculus / α-equivalence)
- name-restriction
  (e.g. as in the π-calculus / structural congruence)

Aim: describe the topos of nominal sets
(a simple reformulation of Schanuel's atomic topos)

as a model for computations on syntactical structures involving

- freshness of names
  (as in "choose a fresh name")
- name-abstraction
  (e.g. as in λ-calculus / α-equivalence)
- name-restriction
  (e.g. as in the π-calculus / structural congruence)

The key idea is equivariance: regard names as atoms and enforce anonymity in binding constructs through invariance under atom-permutations.

Atoms, permutations and actions

- $\mathbb{A} \triangleq$ fixed, countably infinite set, whose elements will be called **atoms**.

Atoms, permutations and actions

- $\mathbb{A} \triangleq$ fixed, countably infinite set, whose elements will be called atoms.
- $\mathbb{G} \triangleq$ group of all finite permutations of $\mathbb{A}$.

Atoms, permutations and actions

- $\mathbb{A} \triangleq$ fixed, countably infinite set, whose elements will be called atoms.
- $\mathbb{G} \triangleq$ group of all finite permutations of $\mathbb{A}$.
- An action of $\mathbb{G}$ on a set X is a function

$$\mathbb{G} \times X \to X \quad \text{written} \quad (\pi, x) \mapsto \pi \cdot x$$

satisfying $\iota \cdot x = x$ and $\pi \cdot (\pi' \cdot x) = (\pi\pi') \cdot x$.

Atoms, permutations and actions

- $\mathbb{A} \triangleq$ fixed, countably infinite set, whose elements will be called atoms.
- $\mathbb{G} \triangleq$ group of all finite permutations of $\mathbb{A}$.
- An action of $\mathbb{G}$ on a set X is a function

$$\mathbb{G} \times X \to X \quad \text{written} \quad (\pi, x) \mapsto \pi \cdot x$$

satisfying $\iota \cdot x = x$ and $\pi \cdot (\pi' \cdot x) = (\pi\pi') \cdot x$.

- $\mathbb{G}$-set $\triangleq$ set X + action of $\mathbb{G}$ on X.

Atoms, permutations and actions

- $\mathbb{A} \triangleq$ fixed, countably infinite set, whose elements will be called atoms.
- $\mathbb{G} \triangleq$ group of all finite permutations of $\mathbb{A}$.
- An action of $\mathbb{G}$ on a set X is a function

$$\mathbb{G} \times X \to X \quad \text{written} \quad (\pi, x) \mapsto \pi \cdot x$$

satisfying $\iota \cdot x = x$ and $\pi \cdot (\pi' \cdot x) = (\pi\pi') \cdot x$.

- $\mathbb{G}$-set $\triangleq$ set X + action of $\mathbb{G}$ on X.
- A function $f : X \to Y$ between $\mathbb{G}$-sets is equivariant if $f(\pi \cdot x) = \pi \cdot (f\, x)$, for all $\pi \in \mathbb{G}$ and $x \in X$.

Languages are $\mathbb{G}$-sets

For example, λ-terms modulo α-equivalence

$$\{t ::= a \mid \lambda a\, t \mid t\, t\}/{=_\alpha}$$

with $\mathbb{G}$-action recursively defined by:

$$\begin{aligned}
\pi \cdot a &=_\alpha \pi(a) \\
\pi \cdot (\lambda a\, t) &=_\alpha \lambda \pi(a)\, (\pi \cdot t) \\
\pi \cdot (t\, t') &=_\alpha (\pi \cdot t)(\pi \cdot t')
\end{aligned}$$

Languages are $\mathbb{G}$-sets

For example, λ-terms modulo α-equivalence

$$\{t ::= a \mid \lambda a\, t \mid t\, t\}/{=_\alpha}$$

with $\mathbb{G}$-action recursively defined by:

$$\begin{aligned}
\pi \cdot a &=_\alpha \pi(a) \\
\pi \cdot (\lambda a\, t) &=_\alpha \lambda \pi(a)\,(\pi \cdot t) \\
\pi \cdot (t\, t') &=_\alpha (\pi \cdot t)(\pi \cdot t')
\end{aligned}$$

N.B. binding and non-binding constructs are treated just the same

Languages are $\mathbb{G}$-sets

For example, λ-terms modulo α-equivalence

$$\{t ::= a \mid \lambda a\, t \mid t\, t\}/{=_\alpha}$$

with $\mathbb{G}$-action recursively defined by:

$$\begin{aligned}
\pi \cdot a &=_\alpha \pi(a) \\
\pi \cdot (\lambda a\, t) &=_\alpha \lambda \pi(a)\, (\pi \cdot t) \\
\pi \cdot (t\, t') &=_\alpha (\pi \cdot t)(\pi \cdot t')
\end{aligned}$$

Lemma: a is not free in t iff $(a\, a') \cdot t =_\alpha t$ holds for all but finitely many atoms a'.

Languages are $\mathbb{G}$-sets

For example, λ-terms modulo α-equivalence

$$\{t ::= a \mid \lambda a\, t \mid t\, t\}/{=_\alpha}$$

with $\mathbb{G}$-action recursively defined by:

$$\begin{aligned}
\pi \cdot a \;&=_\alpha\; \pi(a) \\
\pi \cdot (\lambda a\, t) \;&=_\alpha\; \lambda \pi(a)\, (\pi \cdot t) \\
\pi \cdot (t\, t') \;&=_\alpha\; (\pi \cdot t)(\pi \cdot t')
\end{aligned}$$

Lemma: a is not free in t iff $(a\, a') \cdot t =_\alpha t$ holds for all but finitely many atoms a'.

permutation transposing a and a'

Languages are $\mathbb{G}$-sets

For example, λ-terms modulo α-equivalence

$$\{t ::= a \mid \lambda a\, t \mid t\, t\}/{=_\alpha}$$

with $\mathbb{G}$-action recursively defined by:

$$\begin{aligned} \pi \cdot a &=_\alpha \pi(a) \\ \pi \cdot (\lambda a\, t) &=_\alpha \lambda \pi(a)\, (\pi \cdot t) \\ \pi \cdot (t\, t') &=_\alpha (\pi \cdot t)(\pi \cdot t') \end{aligned}$$

Lemma: a is not free in t iff $(a\, a') \cdot t =_\alpha t$ holds for all but finitely many atoms a'.

So what?

Languages are $\mathbb{G}$-sets

For example, λ-terms modulo α-equivalence

$$\{t ::= a \mid \lambda a\, t \mid t\, t\}/{=_\alpha}$$

with $\mathbb{G}$-action recursively defined by:

$$\begin{aligned} \pi \cdot a &=_\alpha \pi(a) \\ \pi \cdot (\lambda a\, t) &=_\alpha \lambda \pi(a)\, (\pi \cdot t) \\ \pi \cdot (t\, t') &=_\alpha (\pi \cdot t)(\pi \cdot t') \end{aligned}$$

Lemma: a is not free in t iff $(a\, a') \cdot t =_\alpha t$ holds for all but finitely many atoms a'.

So what? Lemma suggests a syntax-independent notion of freshness ("not free in") relation...

Finite support

<u>Definition</u>: a finite set of atoms $\overline{a} \subset \mathbb{A}$ supports an element $x \in X$ of a $\mathbb{G}$-set X if $(a\,a') \cdot x = x$ holds for all $a, a' \in \mathbb{A} - \overline{a}$.

A nominal set is a $\mathbb{G}$-set all of whose elements have a finite support.

Finite support

Definition: a finite set of atoms $\overline{a} \subset \mathbb{A}$ **supports** an element $x \in X$ of a $\mathbb{G}$-set X if $(a\,a') \cdot x = x$ holds for all $a, a' \in \mathbb{A} - \overline{a}$.

A **nominal set** is a $\mathbb{G}$-set all of whose elements have a finite support.

Lemma: If $x \in X$ has a finite support, then it has a smallest one, written $\boxed{supp(x)}$.

E.g. for a λ-term, $supp(t) = \{\text{free variables of } t\}$.

Finite support

<u>Definition</u>: a finite set of atoms $\overline{a} \subset \mathbb{A}$ supports an element $x \in X$ of a $\mathbb{G}$-set X if $(a\, a') \cdot x = x$ holds for all $a, a' \in \mathbb{A} - \overline{a}$.

A nominal set is a $\mathbb{G}$-set all of whose elements have a finite support.

$\mathcal{N}set$ = category of nominal sets and equivariant functions.

$\mathcal{N}set$ is a topos

$\mathcal{N}set$ is a topos

- Terminal object: $\mathbf{1} \triangleq \{0\}$
  action $\pi \cdot 0 = 0$,
  support $supp(0) = \emptyset$.

$\mathcal{N}set$ is a topos

- Terminal object: $\mathbf{1} \triangleq \{\mathbf{0}\}$
  action $\pi \cdot \mathbf{0} = \mathbf{0}$,
  support $supp(\mathbf{0}) = \emptyset$.
- Product: $X \times Y \triangleq \{(x, y) \mid x \in Y \,\&\, y \in Y\}$
  action $\pi \cdot (x, y) \triangleq (\pi \cdot x, \pi \cdot y)$,
  support $supp(x, y) = supp(x) \cup supp(y)$.

$\mathcal{N}set$ is a topos

- Terminal object: $\mathbf{1} \triangleq \{0\}$
  action $\pi \cdot 0 = 0$,
  support $supp(0) = \emptyset$.
- Product: $X \times Y \triangleq \{(x, y) \mid x \in Y \ \& \ y \in Y\}$
  action $\pi \cdot (x, y) \triangleq (\pi \cdot x, \pi \cdot y)$,
  support $supp(x, y) = supp(x) \cup supp(y)$.
- Powerobjects: $P_{\mathrm{fs}}(X) =$ all subsets $S \subseteq X$ that are finitely supported w.r.t. the action given by $\pi \cdot S \triangleq \{\pi \cdot x \mid x \in S\}$. ($\mathcal{N}set$ is boolean.)

$\mathcal{N}set$ is a topos

- Terminal object: $\mathbf{1} \triangleq \{0\}$
  action $\pi \cdot 0 = 0$,
  support $supp(0) = \emptyset$.
- Product: $X \times Y \triangleq \{(x, y) \mid x \in Y \;\&\; y \in Y\}$
  action $\pi \cdot (x, y) \triangleq (\pi \cdot x, \pi \cdot y)$,
  support $supp(x, y) = supp(x) \cup supp(y)$.
- Powerobjects: $P_{\mathrm{fs}}(X)$ = all subsets $S \subseteq X$ that are finitely supported w.r.t. the action given by $\pi \cdot S \triangleq \{\pi \cdot x \mid x \in S\}$. ($\mathcal{N}set$ is boolean.)
- Exponentials: Y^X = all functions from X to Y that are finitely supported w.r.t the action given by $\pi \cdot f \triangleq \lambda x \in X.\, \pi \cdot (f(\pi^{-1} \cdot x))$.

First-order logic (and arithmetic) in $\mathcal{N}set$ is just like for ordinary sets. For example:

- Negation: if $[\![\phi(x)]\!] = S \in P_{\mathrm{fs}}(X)$, then $[\![\neg\phi(x)]\!] = X - S$.

First-order logic (and arithmetic) in $\mathcal{N}set$ is just like for ordinary sets. For example:

- Negation: if $[\![\phi(x)]\!] = S \in P_{\mathrm{fs}}(X)$, then $[\![\neg\phi(x)]\!] = X - S$.

  ($X - S$ is in $P_{\mathrm{fs}}(X)$ because it is supported by any finite $\overline{a}$ that supports S.)

First-order logic (and arithmetic) in $\mathcal{N}set$ is just like for ordinary sets. For example:

- Negation: if $[\![\phi(x)]\!] = S \in P_{\mathrm{fs}}(X)$, then $[\![\neg\phi(x)]\!] = X - S$.

- For all: if $[\![\phi(x, y)]\!] = S \in P_{\mathrm{fs}}(X \times Y)$, then $[\![\forall x.\, \phi(x, y)]\!] = \{y \in Y \mid \forall x \in X.\, (x, y) \in S\}$.

First-order logic (and arithmetic) in $\mathcal{N}set$ is just like for ordinary sets. For example:

- Negation: if $[\![\phi(x)]\!] = S \in P_{\mathrm{fs}}(X)$, then $[\![\neg\phi(x)]\!] = X - S$.
- For all: if $[\![\phi(x, y)]\!] = S \in P_{\mathrm{fs}}(X \times Y)$, then $[\![\forall x.\, \phi(x, y)]\!] = \{y \in Y \mid \forall x \in X.\, (x, y) \in S\}$.

  ($\{y \in Y \mid \forall x \in X.\, (x, y) \in S\}$ is in $P_{\mathrm{fs}}(Y)$, because it is supported by any finite $\overline{a}$ that supports S.)

Higher-order logic in $\mathcal{N}set$ is like higher-order logic for ordinary sets, except that we have to restrict to finitely supported sets and functions when forming powersets and exponentials.

Higher-order logic in $\mathcal{N}set$ is like higher-order logic for ordinary sets, except that we have to restrict to finitely supported sets and functions when forming powersets and exponentials.

For example
Tarski Fixpoint Theorem: for any monotone and finitely supported function Φ from $P_{\mathrm{fs}}(X)$ to itself, the usual least (pre)fixed point

$$\mu(\Phi) \triangleq \bigcap\{S \in P_{\mathrm{fs}}(X) \mid \Phi(S) \subseteq S\}$$

is again finitely supported, hence in $P_{\mathrm{fs}}(X)$.

Higher-order logic in $\mathcal{N}set$ is like higher-order logic for ordinary sets, except that we have to restrict to finitely supported sets and functions when forming powersets and exponentials.

This rules out the use of choice. For example

$$n \mapsto C(n) \triangleq \{S \subseteq \mathbb{A} \mid card(S) = n\}$$

is a finitely (indeed, emptily) supported function from $\mathbb{N}$ to non-empty elements of $P_{\mathrm{fs}}(P_{\mathrm{fs}}(\mathbb{A}))$,

but there is *no* finitely supported function c from $\mathbb{N}$ to $P_{\mathrm{fs}}(\mathbb{A})$ satisfying

$$\forall n \in \mathbb{N}.\ c(n) \in C(n)$$

Atom-abstraction

Nominal set of atom-abstractions, $[\mathbb{A}]X$

$$[\mathbb{A}]\boldsymbol{X} \triangleq (\mathbb{A} \times \boldsymbol{X})/\sim$$

Nominal set of atom-abstractions, $[\mathbb{A}]X$

$$[\mathbb{A}]X \triangleq (\mathbb{A} \times X)/\sim$$

with equivalence relation $(a, x) \sim (a', x')$ given by:

$\boxed{(a\,a'') \cdot x = (a'\,a'') \cdot x' \text{ in } X\text{, for some (or indeed any) } a'' \text{ s.t. } a'' \mathrel{\#} (a, x, a', x')}$

Nominal set of atom-abstractions, $[\mathbb{A}]X$

$$[\mathbb{A}]X \triangleq (\mathbb{A} \times X)/\!\sim$$

with equivalence relation $(a, x) \sim (a', x')$ given by:

$\boxed{(a\,a'') \cdot x = (a'\,a'') \cdot x' \text{ in } X\text{, for some (or indeed any) } a'' \text{ s.t. } a'' \mathbin{\#} (a, x, a', x')}$

given elements $x \in X$ and $y \in Y$ of nominal sets,
we write $x \mathbin{\#} y$ to mean $supp(x) \cap supp(y) = \emptyset$.

Nominal set of atom-abstractions, $[\mathbb{A}]X$

$$[\mathbb{A}]X \triangleq (\mathbb{A} \times X)/{\sim}$$

with equivalence relation $(a, x) \sim (a', x')$ given by:

$\boxed{(a\,a'') \cdot x = (a'\,a'') \cdot x'$ in X, for some (or indeed any) a'' s.t. $a'' \# (a, x, a', x')}$

Write $\boxed{[a]x}$ for the $\sim$-equivalence class of (a, x).

Nominal set of atom-abstractions, $[\mathbb{A}]X$

$$[\mathbb{A}]X \triangleq (\mathbb{A} \times X)/\sim$$

with equivalence relation $(a, x) \sim (a', x')$ given by:

$(a\, a'') \cdot x = (a'\, a'') \cdot x'$ in X, for some (or indeed any) a'' s.t. $a'' \mathrel{\#} (a, x, a', x')$

Write $\boxed{[a]x}$ for the $\sim$-equivalence class of (a, x).

Action: $\pi \cdot [a]x = [\pi(a)](\pi \cdot x)$

(and it follows that $supp([a]x) = supp(x) - \{a\}$).

Atom-abstraction is extremely well-behaved:

equivariant functions $\boldsymbol{Y} \longrightarrow [\mathbb{A}]\boldsymbol{X}$
naturally correspond to equivariant functions
$\{(\boldsymbol{a}, \boldsymbol{y}) \in \mathbb{A} \times \boldsymbol{Y} \mid \boldsymbol{a} \# \boldsymbol{y}\} \longrightarrow \boldsymbol{X}$

equivariant functions $[\mathbb{A}]\boldsymbol{X} \longrightarrow \boldsymbol{Y}$
naturally correspond to equivariant functions
$\boldsymbol{f} : \mathbb{A} \times \boldsymbol{X} \longrightarrow \boldsymbol{Y}$ s.t. $\boldsymbol{a} \# \boldsymbol{f}(\boldsymbol{a}, \boldsymbol{x})$, all $\boldsymbol{a}, \boldsymbol{x}$

$$[\mathbb{A}](\boldsymbol{X} \times \boldsymbol{Y}) \cong [\mathbb{A}]\boldsymbol{X} \times [\mathbb{A}]\boldsymbol{Y}$$

$$[\mathbb{A}](\boldsymbol{X} + \boldsymbol{Y}) \cong [\mathbb{A}]\boldsymbol{X} + [\mathbb{A}]\boldsymbol{Y}$$

$$[\mathbb{A}](\boldsymbol{Y}^{\boldsymbol{X}}) \cong ([\mathbb{A}]\boldsymbol{Y})^{[\mathbb{A}]\boldsymbol{X}}$$

Atom-abstraction gives the non-recursive essence of α-equivalence.

Theorem: $$\{t ::= a \mid \lambda a\, t \mid t\, t\}/{=_\alpha}$$

(quotient of an inductively defined set)

as an object of $\mathcal{N}set$ is isomorphic to

$$\mu X\, (\mathbb{A} + [\mathbb{A}]X + X \times X)$$

(inductively defined nominal set).

Similarly for other languages involving binders.

Atom-abstraction gives the non-recursive essence of α-equivalence.

Theorem:
$$\{t ::= a \mid \lambda a\, t \mid t\, t\}/{=_\alpha}$$
(quotient of an inductively defined set)

as an object of $\mathcal{N}set$ is isomorphic to
$$\mu X\, (\mathbb{A} + [\mathbb{A}]X + X \times X)$$
(inductively defined nominal set).

Similarly for other languages involving binders.

This observation was the starting point for FreshML/Fresh O'Caml—functional programming for nominal sets and equivariant functions (Shinwell, Gabbay & AMP, ICFP'03).

Atom-restriction

π-Calculus restriction, $\nu x.P$

Its reduction $\dfrac{P[a/x] \to P'[a/x] \qquad a \notin fn(P, P')}{\nu x.\, P \to \nu x.\, P'}$

is specific to π-calculus, but structural congruences

(α)	$\nu x.\, P \equiv \nu x'.\, P[x'/x]$	if $x' \notin fn(P)$
(γ)	$\nu x.\, P \equiv P$	if $x \notin fn(P)$
(σ)	$\nu x.\, \nu x'.\, P \equiv \nu x'.\, \nu x.\, P$	
(ε)	$(\nu x.\, P)\vert P' \equiv \nu x.\, (P\vert P')$	if $x \notin fn(P')$

are quite general properties of a "spatial" notion of scope restriction.

(Cf. Caires & Cardelli, *A spatial logic for concurrency*, CONCUR'02.)

π-Calculus restriction, $\nu x.P$

Its reduction $\dfrac{P[a/x] \to P'[a/x] \qquad a \notin fn(P, P')}{\nu x.\, P \to \nu x.\, P'}$

is specific to π-calculus, but structural congruences

$$
\begin{array}{lll}
(\alpha) & \nu x.\, P \equiv \nu x'.\, P[x'/x] & \text{if } x' \notin fn(P) \\
(\gamma) & \nu x.\, P \equiv P & \text{if } x \notin fn(P) \\
(\sigma) & \nu x.\, \nu x'.\, P \equiv \nu x'.\, \nu x.\, P & \\
(\varepsilon) & (\nu x.\, P)|P' \equiv \nu x.\, (P|P') & \text{if } x \notin fn(P')
\end{array}
$$

are quite general properties of a "spatial" notion of scope restriction.

Aim: a construct in nominal sets satisfying (α)–(σ) that allows us to discuss scope extrusion (ε) in a syntax-independent way.

Nominal restriction structure, (R, ρ)

is given by a nominal set R and an equivariant function $\quad \rho : [\mathbb{A}]R \to R$

satisfying $\begin{cases} a \# r \;\Rightarrow\; \rho[a]r = r \\ \rho[a]\rho[a']r = \rho[a']\rho[a]r \end{cases}$

Nominal restriction structure, (R, ρ)

is given by a nominal set R and an equivariant function $\rho : [\mathbb{A}]R \to R$

satisfying $\begin{cases} a \# r \;\Rightarrow\; \rho[a]r = r \\ \rho[a]\rho[a']r = \rho[a']\rho[a]r \end{cases}$

(α) $\nu x. P \equiv \nu x'. P[x'/x]$ if $x' \notin fn(P)$

Nominal restriction structure, (R, ρ)

is given by a nominal set $\boldsymbol{R}$ and an equivariant function $\boldsymbol{\rho : [\mathbb{A}]R \longrightarrow R}$

satisfying $\begin{cases} \boldsymbol{a \mathrel{\#} r \;\Rightarrow\; \rho[a]r = r} \\ \boldsymbol{\rho[a]\rho[a']r = \rho[a']\rho[a]r} \end{cases}$

(γ) $\nu x.\, P \equiv P$ if $x \notin fn(P)$

Nominal restriction structure, (R, ρ)

is given by a nominal set R and an equivariant function $\quad \rho : [\mathbb{A}]R \to R$

satisfying $\begin{cases} a \# r \;\Rightarrow\; \rho[a]r = r \\ \rho[a]\rho[a']r = \rho[a']\rho[a]r \end{cases}$

$(\sigma)\;\; \nu x.\nu x'.P \equiv \nu x'.\nu x.P$

Nominal restriction structure, (R, ρ)

is given by a nominal set R and an equivariant function $\quad \rho : [\mathbb{A}]R \longrightarrow R$

satisfying $\begin{cases} a \# r \;\Rightarrow\; \rho[a]r = r \\ \rho[a]\rho[a']r = \rho[a']\rho[a]r \end{cases}$

Theorem: every nominal set X possesses a freely generated restriction structure:

Nominal restriction structure, (R, ρ)

is given by a nominal set R and an equivariant function $\quad \rho : [\mathbb{A}]R \longrightarrow R$

satisfying $\begin{cases} a \# r \ \Rightarrow \ \rho[a]r = r \\ \rho[a]\rho[a']r = \rho[a']\rho[a]r \end{cases}$

<u>Theorem</u>: every nominal set X possesses a freely generated restriction structure:

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "\forall"'] & \nu X & (\nu X, \rho_X) \\
& R & (R, \rho)
\end{tikzcd}$$

Nominal restriction structure, (R, ρ)

is given by a nominal set R and an equivariant function $\rho : [\mathbb{A}]R \longrightarrow R$

satisfying $\begin{cases} a \# r \;\Rightarrow\; \rho[a]r = r \\ \rho[a]\rho[a']r = \rho[a']\rho[a]r \end{cases}$

<u>Theorem</u>: every nominal set X possesses a freely generated restriction structure:

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[dr, "\forall"'] & \nu X \arrow[d, dashed, "\exists!"] & (\nu X, \rho_X) \arrow[d, dashed, "\text{preserving restriction}"] \\ & R & (R, \rho) \end{tikzcd}$$

Construction of νX

$$\nu X \triangleq (X \times P_{\mathrm{fin}}\mathbb{A})/{\sim}$$

Construction of νX

$$\nu X \triangleq (X \times P_{\mathrm{fin}}\mathbb{A})/\sim$$

finite sets $\overline{a}$ of atoms

Construction of νX

$$\nu X \triangleq (X \times P_{\mathrm{fin}}\mathbb{A})/\!\sim$$

with equivalence relation $(x, \overline{a}) \sim (x', \overline{a}')$ given by:

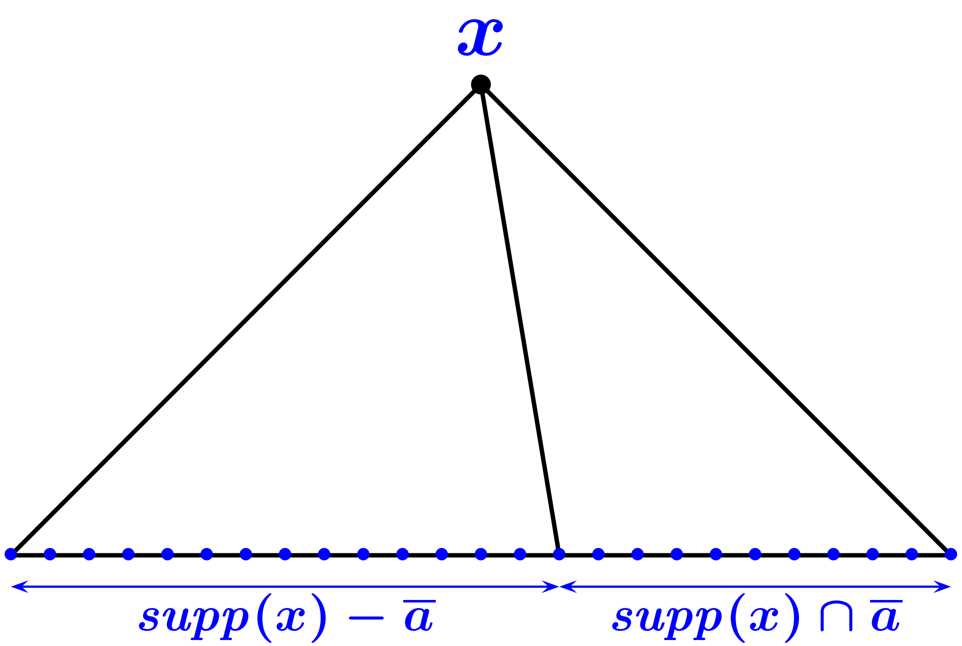
$$\boxed{\begin{array}{c} supp(x) - \overline{a} = supp(x') - \overline{a}' \text{ and } \pi \cdot x = x' \\ \text{for some } \pi \in \mathbb{G} \text{ with } \pi \# supp(x) - \overline{a} \end{array}}$$

Construction of νX

$$\nu X \triangleq (X \times P_{\mathrm{fin}}\mathbb{A})/\sim$$

with equivalence relation $(x, \overline{a}) \sim (x', \overline{a}')$ given by:

$$\boxed{\begin{array}{c} supp(x) - \overline{a} = supp(x') - \overline{a}' \text{ and } \pi \cdot x = x' \\ \text{for some } \pi \in \mathbb{G} \text{ with } \pi \mathbin{\#} supp(x) - \overline{a} \end{array}}$$

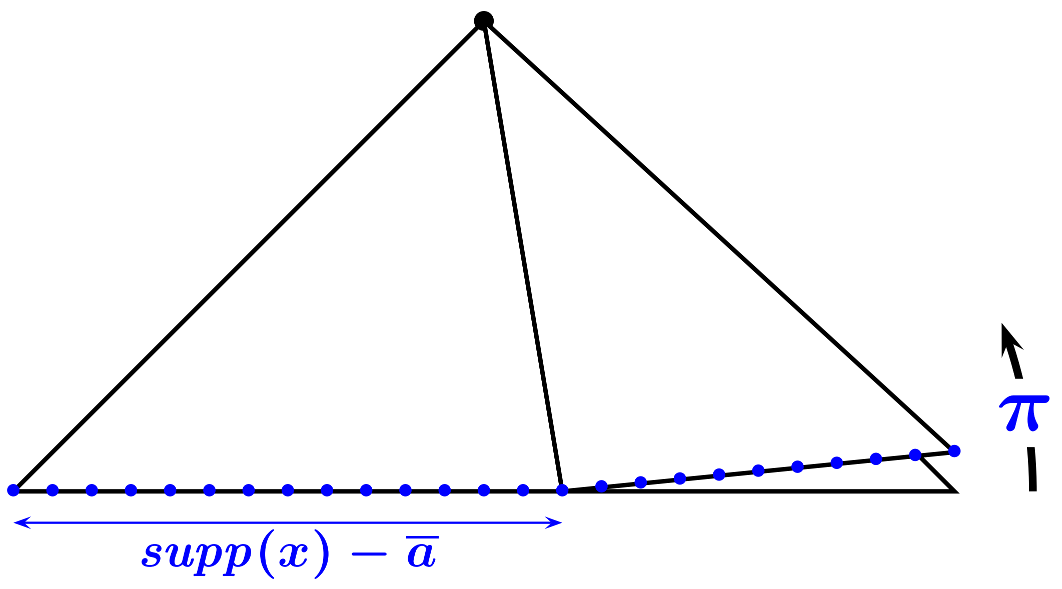


Construction of νX

$$\nu X \triangleq (X \times P_{\mathrm{fin}}\mathbb{A})/\sim$$

with equivalence relation $(x, \overline{a}) \sim (x', \overline{a}')$ given by:

$$\boxed{\begin{array}{c} supp(x) - \overline{a} = supp(x') - \overline{a}' \text{ and } \pi \cdot x = x' \\ \text{for some } \pi \in \mathbb{G} \text{ with } \pi \# supp(x) - \overline{a} \end{array}}$$

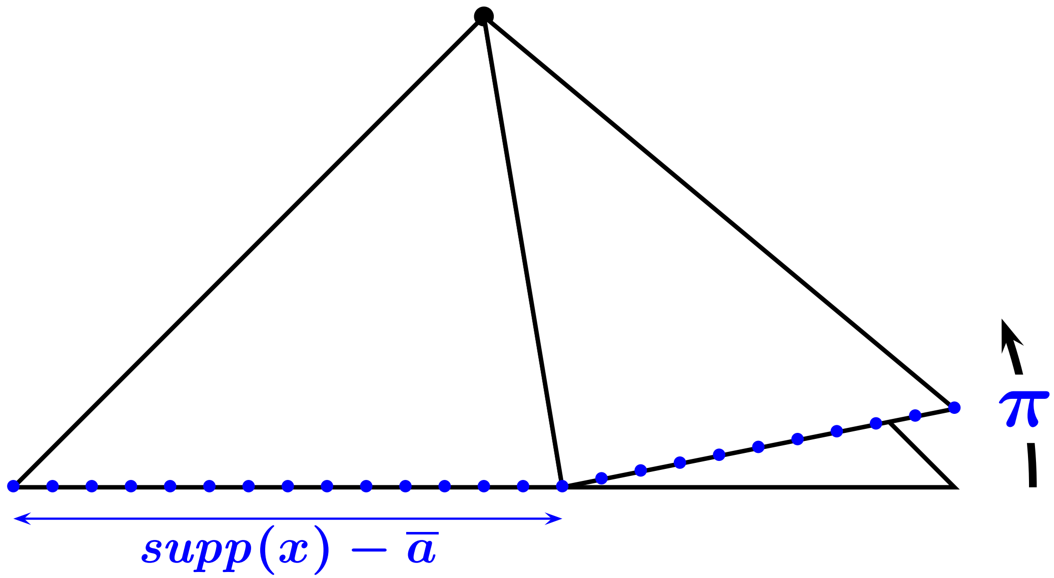


Construction of νX

$$\nu X \triangleq (X \times P_{\mathrm{fin}}\mathbb{A})/\sim$$

with equivalence relation $(x, \overline{a}) \sim (x', \overline{a}')$ given by:

$$\boxed{\begin{array}{c} supp(x) - \overline{a} = supp(x') - \overline{a}' \text{ and } \pi \cdot x = x' \\ \text{for some } \pi \in \mathbb{G} \text{ with } \pi \# supp(x) - \overline{a} \end{array}}$$

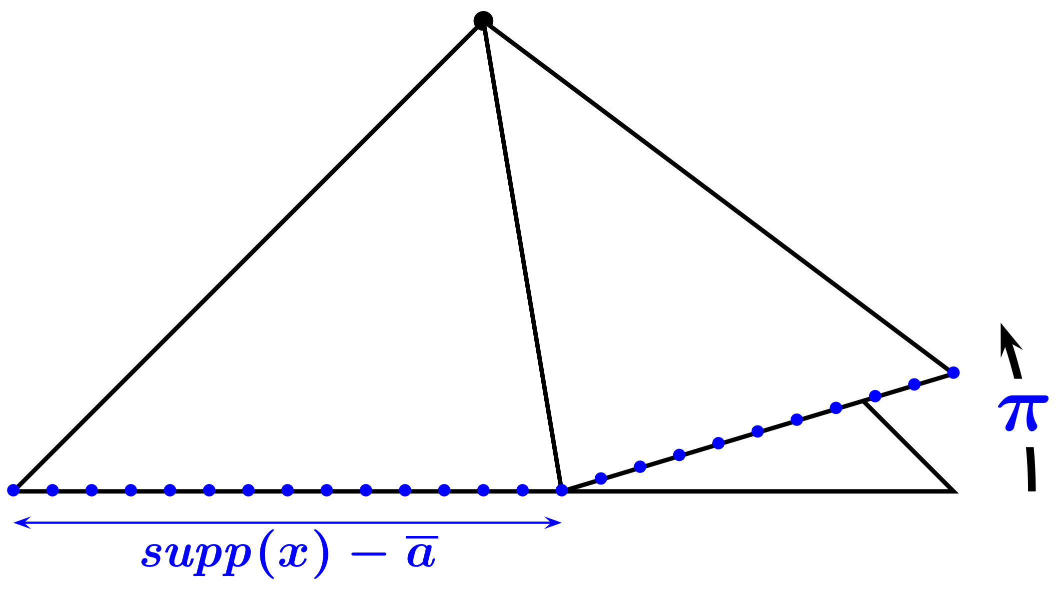


Construction of νX

$$\nu X \triangleq (X \times P_{\mathrm{fin}}\mathbb{A})/\!\sim$$

with equivalence relation $(x, \overline{a}) \sim (x', \overline{a}')$ given by:

$$\boxed{\begin{array}{c} supp(x) - \overline{a} = supp(x') - \overline{a}' \text{ and } \pi \cdot x = x' \\ \text{for some } \pi \in \mathbb{G} \text{ with } \pi \# supp(x) - \overline{a} \end{array}}$$

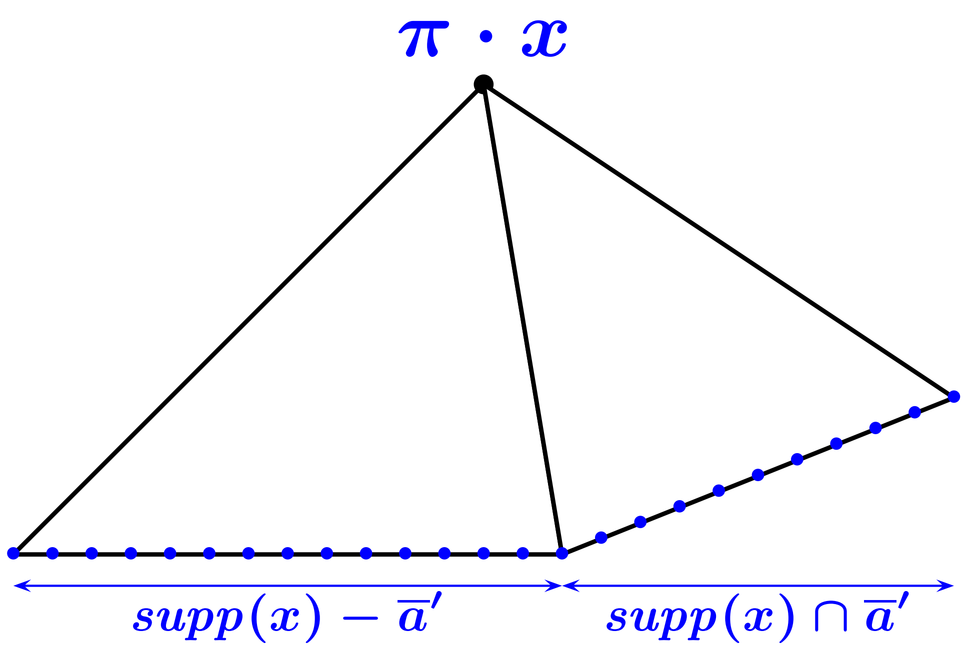


Construction of νX

$$\nu X \triangleq (X \times P_{\mathrm{fin}}\mathbb{A})/\!\sim$$

with equivalence relation $(x, \overline{a}) \sim (x', \overline{a}')$ given by:

$$\boxed{\begin{array}{c} supp(x) - \overline{a} = supp(x') - \overline{a}' \text{ and } \pi \cdot x = x' \\ \text{for some } \pi \in \mathbb{G} \text{ with } \pi \mathrel{\#} supp(x) - \overline{a} \end{array}}$$



Construction of νX

$$\nu X \triangleq (X \times P_{\mathrm{fin}}\mathbb{A})/{\sim}$$

with equivalence relation $(x, \overline{a}) \sim (x', \overline{a}')$ given by:

$$\boxed{\begin{array}{c} supp(x) - \overline{a} = supp(x') - \overline{a}' \text{ and } \pi \cdot x = x' \\ \text{for some } \pi \in \mathbb{G} \text{ with } \pi \# supp(x) - \overline{a} \end{array}}$$

Write $\boxed{x \diagdown \overline{a}}$ for the $\sim$-equivalence class of $(x, \overline{a})$.

Construction of νX

$$\nu X \triangleq (X \times P_{\mathrm{fin}}\mathbb{A})/{\sim}$$

with equivalence relation $(x, \overline{a}) \sim (x', \overline{a}')$ given by:

$$\boxed{\begin{array}{c} supp(x) - \overline{a} = supp(x') - \overline{a}' \text{ and } \pi \cdot x = x' \\ \text{for some } \pi \in \mathbb{G} \text{ with } \pi \mathrel{\#} supp(x) - \overline{a} \end{array}}$$

Write $\boxed{x \setminus \overline{a}}$ for the $\sim$-equivalence class of $(x, \overline{a})$.

$\mathbb{G}$-action is: $\pi \cdot (x \setminus \overline{a}) \triangleq (\pi \cdot x) \setminus \{\pi(a) \mid a \in \overline{a}\}$
(and it follows that $supp(x \setminus \overline{a}) = supp(x) - \overline{a}$).

Construction of νX

$$\nu X \triangleq (X \times P_{\mathrm{fin}}\mathbb{A})/\!\sim$$

with equivalence relation $(x, \overline{a}) \sim (x', \overline{a}')$ given by:

$$\boxed{\begin{array}{c} \mathit{supp}(x) - \overline{a} = \mathit{supp}(x') - \overline{a}' \text{ and } \pi \cdot x = x' \\ \text{for some } \pi \in \mathbb{G} \text{ with } \pi \mathrel{\#} \mathit{supp}(x) - \overline{a} \end{array}}$$

Write $\boxed{x \diagdown \overline{a}}$ for the $\sim$-equivalence class of $(x, \overline{a})$.

$\mathbb{G}$-action is: $\pi \cdot (x \diagdown \overline{a}) \triangleq (\pi \cdot x) \diagdown \{\pi(a) \mid a \in \overline{a}\}$
(and it follows that $\mathit{supp}(x \diagdown \overline{a}) = \mathit{supp}(x) - \overline{a}$).

Restriction operation $\rho_X : [\mathbb{A}]\nu X \to \nu X$ is:

$$\rho_X[a](x \diagdown \overline{a}) \triangleq x \diagdown (\{a\} \cup \overline{a})$$

Construction of νX

$$\nu X \triangleq (X \times P_{\mathrm{fin}}\mathbb{A})/{\sim}$$

with equivalence relation $(x, \overline{a}) \sim (x', \overline{a}')$ given by:

$$\boxed{\begin{array}{c} supp(x) - \overline{a} = supp(x') - \overline{a}' \text{ and } \pi \cdot x = x' \\ \text{for some } \pi \in \mathbb{G} \text{ with } \pi \mathrel{\#} supp(x) - \overline{a} \end{array}}$$

Write $\boxed{x \setminus \overline{a}}$ for the $\sim$-equivalence class of $(x, \overline{a})$.

$\mathbb{G}$-action is: $\pi \cdot (x \setminus \overline{a}) \triangleq (\pi \cdot x) \setminus \{\pi(a) \mid a \in \overline{a}\}$ (and it follows that $supp(x \setminus \overline{a}) = supp(x) - \overline{a}$).

Insertion operation $\eta_X : X \to \nu X$ is:

$$\eta_X(x) \triangleq x \setminus \emptyset$$

Abstraction from restriction

The equivalence relation used to construct νX seems quite similar to that used to construct $[\mathbb{A}]X$. What is the relationship?

Abstraction from restriction

The equivalence relation used to construct $\boldsymbol{\nu X}$ seems quite similar to that used to construct $[\mathbb{A}]\boldsymbol{X}$. What is the relationship?

<u>Lemma</u>: The subset of $\boldsymbol{\nu}(\mathbb{A} \times \boldsymbol{X})$ given by

$$\{(\boldsymbol{a}, \boldsymbol{x}) \setminus \{\boldsymbol{a}\} \mid \boldsymbol{a} \in \mathbb{A} \ \& \ \boldsymbol{x} \in \boldsymbol{X}\}$$

is isomorphic to $[\mathbb{A}]\boldsymbol{X}$ via the correspondence

$$(\boldsymbol{a}, \boldsymbol{x}) \setminus \{\boldsymbol{a}\} \longleftrightarrow [\boldsymbol{a}]\boldsymbol{x}$$

Scope extrusion

For π-calculus, symmetry of $(-)|(-)$ plus

$$(\varepsilon) \quad (\nu x.\, P)|P' \equiv \nu x.\, (P|P') \quad \text{if } x \notin fn(P')$$

says operation $(-)|(-)$ on $\{\pi\text{-processes}\}/\equiv$ is a

bimorphism of restriction structures:
equivariant function $(-)|(-) : R_1 \times R_2 \to R_3$
satisfying
$$\begin{cases} (\rho_1[a]r)|r' = \rho_3[a](r|r') & \text{if } a \# r' \\ r|(\rho_2[a]r') = \rho_3[a](r|r') & \text{if } a \# r \end{cases}$$

Scope extrusion

For π-calculus, symmetry of $(-)|(-)$ plus

$$(\varepsilon) \quad (\nu x.\, P)|P' \equiv \nu x.\, (P|P') \quad \text{if } x \notin fn(P')$$

says operation $(-)|(-)$ on $\{\pi\text{-processes}\}/\equiv$ is a

bimorphism of restriction structures: equivariant function $(-)|(-) : R_1 \times R_2 \to R_3$ satisfying

$$\begin{cases} (\rho_1[a]r)|r' = \rho_3[a](r|r') & \text{if } a \# r' \\ r|(\rho_2[a]r') = \rho_3[a](r|r') & \text{if } a \# r \end{cases}$$

Theorem: Given R_1 & R_2, there is a universal bimorphism $R_1 \times R_2 \longrightarrow R_1 \oplus R_2$

Scope extrusion

For π-calculus, symmetry of $(-)|(-)$ plus

$$(\varepsilon) \quad (\nu x.\, P)|P' \equiv \nu x.\, (P|P') \quad \text{if } x \notin fn(P')$$

says operation $(-)|(-)$ on $\{\pi\text{-processes}\}/\!\equiv$ is a

bimorphism of restriction structures:
equivariant function $(-)|(-) : R_1 \times R_2 \to R_3$

satisfying $\begin{cases} (\rho_1[a]r)|r' = \rho_3[a](r|r') & \text{if } a \# r' \\ r|(\rho_2[a]r') = \rho_3[a](r|r') & \text{if } a \# r \end{cases}$

<u>Theorem</u>: Given R_1 & R_2, there is a universal bimorphism

$$\begin{tikzcd} R_1 \times R_2 \arrow[rr] \arrow[drr, "\forall \text{ bimorphisms}"'] & & R_1 \varobar R_2 \\ & & R_3 \end{tikzcd}$$

Scope extrusion

For π-calculus, symmetry of $(-)|(-)$ plus

$$(\varepsilon) \quad (\nu x.\, P)|P' \equiv \nu x.\, (P|P') \quad \text{if } x \notin fn(P')$$

says operation $(-)|(-)$ on $\{\pi\text{-processes}\}/\equiv$ is a

bimorphism of restriction structures:
equivariant function $(-)|(-) : R_1 \times R_2 \to R_3$
satisfying
$$\begin{cases} (\rho_1[a]r)|r' = \rho_3[a](r|r') & \text{if } a \# r' \\ r|(\rho_2[a]r') = \rho_3[a](r|r') & \text{if } a \# r \end{cases}$$

Theorem: Given R_1 & R_2, there is a universal bimorphism

$$\begin{tikzcd} R_1 \times R_2 \arrow[rr] \arrow[drr, "\forall \text{ bimorphisms}"'] & & R_1 \oplus R_2 \arrow[d, dashed, "\exists! \text{ morphism}"] \\ & & R_3 \end{tikzcd}$$

????

Is there a (useful!) initial algebra semantics for "spatial" calculi/logics (e.g. π, ambients, TQL, ...) using restriction structures in nominal sets?

- Use $[\mathbb{A}](-)$ for domain of name-binding operators.
- Use $- \oplus -$ for domain of binary, spatial operators.
- Use $- \times -$ for domain of binary, non-spatial operators.

(Cf. presheaf semantics of π-calculus: Stark (LICS'96), Fiore-Moggi-Sangiorgi (LICS'96).)

Equations + freshness constraints = ?

In $\mathcal{N}set$ we can model equational theories conditioned by freshness, e.g.:

$$
\begin{aligned}
a' \# x &\vdash R(a, x) = R(a', (a\, a') \cdot x) \\
&\vdash R(a, R(a' x)) = R(a', R(a, x)) \\
a \# x &\vdash R(a, x) = x
\end{aligned}
$$

Equations + freshness constraints = ?

In $\mathcal{N}set$ we can model equational theories conditioned by freshness, e.g.:

$$
\begin{aligned}
a'\#x &\vdash R(a,x) = R(a',(a\,a')\cdot x)\\
&\vdash R(a,R(a'x)) = R(a',R(a,x))\\
a\#x &\vdash R(a,x) = x
\end{aligned}
$$

What is the categorical logic of this kind of theory? Is there a notion of classifying category (internal to $\mathcal{N}set$) for this kind of theory within $\mathcal{N}set$?

(Cf. Urban, Gabbay & AMP, *Nominal Unification*, TCS to appear.)

Dynamic allocation monads

$X \mapsto (\nu X, \rho_X)$ reflects nominal sets into nominal restriction structures. So $\nu(-)$ is a monad on $\mathcal{N}set$.

In fact the explicit construction of νX shows that $\nu(-)$ corresponds to one of Moggi's dynamic allocation monads on a functor-category, used for semantics of name creation.

Dynamic allocation monads

$X \mapsto (\nu X, \rho_X)$ reflects nominal sets into nominal restriction structures. So $\nu(-)$ is a monad on $\mathcal{N}set$.

In fact the explicit construction of νX shows that $\nu(-)$ corresponds to one of Moggi's dynamic allocation monads on a functor-category, used for semantics of name creation.

Abramsky-Ghica-Murawski-Ong-Stark (LICS'04) make use of nominal sets and $\nu(-)$ to give a fully abstract game semantics for the Pitts-Stark ν-calculus.

Dynamic allocation monads

$X \mapsto (\nu X, \rho_X)$ reflects nominal sets into nominal restriction structures. So $\nu(-)$ is a monad on $\mathcal{N}set$.

In fact the explicit construction of νX shows that $\nu(-)$ corresponds to one of Moggi's dynamic allocation monads on a functor-category, used for semantics of name creation.

Abramsky-Ghica-Murawski-Ong-Stark (LICS'04) make use of nominal sets and $\nu(-)$ to give a fully abstract game semantics for the Pitts-Stark ν-calculus.

What about old-fashioned denotational semantics, using domains?

Dynamic allocation monads

$X \mapsto (\nu X, \rho_X)$ reflects nominal sets into nominal restriction structures. So $\nu(-)$ is a monad on $\mathcal{N}set$.

In fact the explicit construction of νX shows that $\nu(-)$ corresponds to one of Moggi's dynamic allocation monads on a functor-category, used for semantics of name creation.

Abramsky-Ghica-Murawski-Ong-Stark (LICS'04) make use of nominal sets and $\nu(-)$ to give a fully abstract game semantics for the Pitts-Stark ν-calculus.

What about old-fashioned denotational semantics, using domains? I'm glad you asked...

Domain theory in $\mathcal{N}set$

ω-Cpos in $\mathcal{N}set$

Conventional domain theory in $\mathcal{N}set$, up to and including the construction of recursively defined domains, is no harder than in $\mathcal{S}et$.

N.B. an internal ω-chain is just an increasing sequence $d_0 \sqsubseteq d_1 \sqsubseteq \cdots$ for which there is a single finite set of atoms supporting all the d_n.
So ω-cpos in $\mathcal{N}set$ may be incomplete externally—e.g. $(P_{\mathrm{fin}}(\mathbb{A}), \subseteq)$.

ω-Cpos in $\mathcal{N}set$

<u>Problem</u>:
Unlike $[\mathbb{A}]\boldsymbol{D}$, the free restriction structure $\boldsymbol{\nu D}$ is not always an ω-cpo in $\mathcal{N}\boldsymbol{set}$ even if $\boldsymbol{D}$ is.
(Partly explains why dynamic allocation monads on functor categories have had limited application.)

ω-Cpos in $\mathcal{N}set$

Problem:

Unlike $[\mathbb{A}]D$, the free restriction structure νD is not always an ω-cpo in $\mathcal{N}set$ even if D is.
(Partly explains why dynamic allocation monads on functor categories have had limited application.)

Solution:

Continuous function domain $D \to \{\bot, \top\}$ possesses a restriction structure; hence we can use the continuation monad $((-) \to \{\bot, \top\}) \to \{\bot, \top\}$ to give a denotational semantics of dynamic allocation + fixpoint recursion.

See: Shinwell-Pitts, "*On a Monadic Semantics for Freshness*" (APPSEM'04).

Conclusion

Nominal sets provide a model of

- restriction & anonymity (via permutation-invariance)
- name-abstraction & implicit dependence on parameters (via the notion of "support")
- function-abstraction & explicit dependence on parameters (via exponentials)
- dynamic allocation of fresh names (via the notion of support again)

that is pretty, pretty simple, and pretty rich in interesting properties.

We are only at the beginning of the computational consequences of taking this model seriously.

Thanks

James Cheney (Cornell), Jamie Gabbay (INRIA),
Mark Shinwell & Christian Urban (Cambridge).

Further info

`www.cl.cam.ac.uk/users/amp12/freshml/`