An Invitation to Nominal Domain Theory

Andrew Pitts

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Nominal domain theory

is domain theory in the internal HO logic of the category of nominal sets, $\boldsymbol{\mathcal{N}om}$.

Nominal domain theory

is domain theory in the internal HO logic of the category of nominal sets, $\mathcal{N}\!\mathit{om}$.

- $\mathcal{N}\!\mathit{om}$ is "just" a concrete presentation of Schanuel's atomic topos, oriented towards the syntax-independent, mathematical model of name-binding and freshness that it supports.

Nominal domain theory

is domain theory in the internal HO logic of the category of nominal sets, $\mathcal{N}om$.

- $\mathcal{N}om$ is "just" a concrete presentation of Schanuel's atomic topos, oriented towards the syntax-independent, mathematical model of name-binding and freshness that it supports.

as in "choose a fresh name"

Nominal domain theory

is domain theory in the internal HO logic of the category of nominal sets, $\mathcal{N}om$.

- $\mathcal{N}om$ is "just" a concrete presentation of Schanuel's atomic topos, oriented towards the syntax-independent, mathematical model of name-binding and freshness that it supports.
- $\mathcal{N}om$ was one of two independent solutions circa 1999 for giving syntax/α an initial algebra semantics.

Nominal domain theory

is domain theory in the internal HO logic of the category of nominal sets, $\mathcal{N}om$.

- $\mathcal{N}om$ is "just" a concrete presentation of Schanuel's atomic topos, oriented towards the syntax-independent, mathematical model of name-binding and freshness that it supports.
- ~~$\mathcal{N}om$~~ was one of two independent solutions circa 1999 for giving syntax/α an initial algebra semantics.

FM-set theory

see Proc. LICS 1999

Atoms, permutations and actions

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

$$(\pi, x) \in \mathbb{G} \times X \mapsto \pi \cdot x \in X$$

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

Finite support

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

Finite support

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

permutation that
transposes a and a'

Finite support

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

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

Finite support

A finite set of atoms $\overline{a} \subset \mathbb{A}$ supports $x \in X$ if $(a\ a') \cdot x = x$, 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.

Motivating example: for a λ-term t

$$supp(t) = \{\text{free variables of } t\}$$

Finite support

A finite set of atoms $\overline{a} \subset \mathbb{A}$ supports $x \in X$ if $(a\ a') \cdot x = x$, 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.

Another example, nominal set of atoms: $\mathbb{A}$ + action $\pi \cdot a \triangleq \pi(a)$, for which

$$supp(a) = \{a\}$$

Finite support

A finite set of atoms $\overline{a} \subset \mathbb{A}$ supports $x \in X$ if $(a\ a') \cdot x = x$, 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.

Notation: $\mathcal{N}om$ = category of nominal sets
and equivariant functions.

Finite support

A finite set of atoms $\overline{a} \subset \mathbb{A}$ supports $x \in X$ if $(a\ a') \cdot x = x$, 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.

Notation: $\mathcal{N}om$ = category of nominal sets and equivariant functions.

functions $f : X \to Y$ between $\mathbb{G}$-sets satisfying

$$f(\pi \cdot x) = \pi \cdot (f x)$$

$\mathcal{N}om$ is a topos

- Finite limits and NNO: created by forgetful functor $\mathcal{N}om \longrightarrow \mathcal{S}et$.
- Powerobjects: $P_{\mathrm{fs}}(X) =$ all subsets $S \subseteq X$ that are finitely supported w.r.t. the action given by $\pi \cdot S \triangleq \{\pi \cdot x \mid x \in S\}$.

$\mathcal{Nom}$ is a topos

- Finite limits and NNO: created by forgetful functor $\mathcal{Nom} \to \mathcal{Set}$.
- Powerobjects: $P_{\mathrm{fs}}(X)$ = all subsets $S \subseteq X$ that are finitely supported w.r.t. the action given by $\pi \cdot S \triangleq \{\pi \cdot x \mid x \in S\}$.

not every subset is finitely supported.
E.g. $S \subseteq \mathbb{A}$ is f.s. iff either S or $\mathbb{A} - S$ is finite

$\mathcal{N}om$ is a topos

- Finite limits and NNO: created by forgetful functor $\mathcal{N}om \longrightarrow \mathcal{S}et$.
- Powerobjects: $P_{\mathrm{fs}}(X)$ = all subsets $S \subseteq X$ that are finitely supported w.r.t. the action given by $\pi \cdot S \triangleq \{\pi \cdot x \mid x \in S\}$.
- Exponentials: Y^X = all functions from X to Y that are finitely supported w.r.t the action given by $(\pi \cdot f)(x) = \pi \cdot (f(\pi^{-1} \cdot x))$.

First-order logic (and arithmetic) in $\mathcal{N}om$ is just like for $\mathcal{S}et$. For example:

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

$\mathrm{supp}(X - S) = \mathrm{supp}(S)$

First-order logic (and arithmetic) in $\mathcal{N}om$ is just like for $\mathcal{S}et$. 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 $[\![\forall x.\,\phi(x,y)]\!] = \{\underbrace{y \in Y \mid \forall x \in X.\,(x,y) \in S}_{\text{the support of this is contained in } \mathrm{supp}(S)}\}$.

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

Higher-order logic in $\mathcal{N}om$ is like higher-order logic in $\mathcal{S}et$, except that we have to restrict to finitely supported sets and functions when forming powersets and exponentials—rules out some uses 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

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

Nominal domains

Naïve domain theory:

$$\text{domain} \quad = \quad \begin{array}{l}\omega\text{-chain complete poset} \\ \text{with least element (cppo)}\end{array}$$

interpreted in internal HO logic of $\mathcal{N}om$.

Nominal domains

$$\mathcal{N}dom \triangleq cppo(\mathcal{N}om)$$

Objects $(D, \sqsubseteq, \cdot)$

- $(D, \sqsubseteq)$ poset with $\perp$
- $(D, \cdot)$ nominal set
- $\sqsubseteq$ is equivariant wrt $\cdot$
- every finitely supported ω-chain $d_0 \sqsubseteq d_1 \sqsubseteq \cdots$ in D has a lub

Nominal domains

$$\mathcal{N}dom \triangleq cppo(\mathcal{N}om)$$

Objects $(D, \sqsubseteq, \cdot)$

- $(D, \sqsubseteq)$ poset with $\perp$
- $(D, \cdot)$ nominal set
- $\sqsubseteq$ is equivariant wrt $\cdot$
- every finitely supported ω-chain $d_0 \sqsubseteq d_1 \sqsubseteq \cdots$ in D has a lub

$d \sqsubseteq d' \Rightarrow \pi \cdot d \sqsubseteq \pi \cdot d'$

N.B. this implies that $\pi \cdot \perp = \perp$

Nominal domains

$$\mathcal{N}dom \triangleq cppo(\mathcal{N}om)$$

Objects $(D, \sqsubseteq, \cdot)$

- $(D, \sqsubseteq)$ poset with $\perp$
- $(D, \cdot)$ nominal set
- $\sqsubseteq$ is equivariant wrt $\cdot$
- every finitely supported ω-chain $d_0 \sqsubseteq d_1 \sqsubseteq \cdots$ in D has a lub

so $(D, \sqsubseteq)$ may be incomplete externally — e.g. $(P_{fin}(\mathbb{A}), \subseteq)$

chain is f.s. iff there's a single finite set of atoms supporting all the d_n

Nominal domains

$$\mathcal{N}dom \triangleq cppo(\mathcal{N}om)$$

Morphisms $f : (D, \sqsubseteq, \cdot) \multimap (D' \sqsubseteq, \cdot)$

- f is monotone and strict
- f is equivariant
- f preserves lubs of finitely supported ω-chains.

Nominal domains

$$\mathcal{N}dom \triangleq cppo(\mathcal{N}om)$$

Recursively defined objects

Minimal invariants $\mu(F)$ for locally continuous functors

$$F : \mathcal{N}dom^{op} \times \mathcal{N}dom \to \mathcal{N}dom$$

exist via the usual limit-colimit construction:

Nominal domains

$$\mathcal{N}dom \triangleq cppo(\mathcal{N}om)$$

Recursively defined objects

Minimal invariants $\mu(F)$ for locally continuous functors

$$F : \mathcal{N}dom^{op} \times \mathcal{N}dom \longrightarrow \mathcal{N}dom$$

exist via the usual limit-colimit construction:

= enriched over Cpo($\mathcal{N}$om)

Nominal domains

$$\mathcal{N}dom \triangleq cppo(\mathcal{N}om)$$

Recursively defined objects

Minimal invariants $\mu(F)$ for locally continuous functors

$$F : \mathcal{N}dom^{op} \times \mathcal{N}dom \to \mathcal{N}dom$$

exist via the usual limit-colimit construction:

$\mu(F)$ consists of compatible and finitely supported sequences $(d_n \in F^{(n)} \mid n < \omega)$,

where $\begin{cases} F^{(0)} & \triangleq \emptyset_\perp \\ F^{(n+1)} & \triangleq F(F^{(n)}, F^{(n)}) \end{cases}$

Example: dynamic allocation

Untyped Pitts-Stark ν-calculus :

$$\begin{array}{llll} \text{Values} & v & ::= & x \mid a \mid \lambda x\, e \\ \text{Expressions} & e & ::= & v \mid \texttt{new} \mid v\, v \mid \texttt{let}\, x = e\, \texttt{in}\, e \mid \\ & & & \texttt{if}\, v = v\, \texttt{then}\, e\, \texttt{else}\, e \end{array}$$

Example: dynamic allocation

Untyped Pitts-Stark ν-calculus :

names

Values	v	$::=$	$x \mid a \mid \lambda x\, e$
Expressions	e	$::=$	$v \mid \mathtt{new} \mid v\, v \mid \mathtt{let}\ x = e\ \mathtt{in}\ e \mid$
			$\mathtt{if}\ v = v\ \mathtt{then}\ e\ \mathtt{else}\ e$

dynamically generated fresh name

branch on name equality

Example: dynamic allocation

Untyped Pitts-Stark ν-calculus + Felleisen-style SOS:

Values	v	$::=$	$x \mid a \mid \lambda x\, e$
Expressions	e	$::=$	$v \mid \mathtt{new} \mid v\, v \mid \mathtt{let}\, x = e\, \mathtt{in}\, e \mid$
			$\mathtt{if}\, v = v\, \mathtt{then}\, e\, \mathtt{else}\, e$
Frame-stacks	s	$::=$	$\mathtt{id} \mid s \circ (\lambda x\, e)$

Termination relation $\langle s, e \rangle{\downarrow}$ between closed frame-stacks and closed expressions inductively defined by

$$\frac{}{\langle \mathtt{id}, v \rangle{\downarrow}} \qquad \frac{\langle s, e[v/x] \rangle{\downarrow}}{\langle s \circ (\lambda x\, e), v \rangle{\downarrow}} \qquad \frac{\langle s, a \rangle{\downarrow} \qquad a \notin s}{\langle s, \mathtt{new} \rangle{\downarrow}}$$

etc.

Example: dynamic allocation

Untyped Pitts-Stark ν-calculus :

$$
\begin{array}{llll}
\text{Values} & v & ::= & x \mid a \mid \lambda x\, e \\
\text{Expressions} & e & ::= & v \mid \mathtt{new} \mid v\, v \mid \mathtt{let}\, x = e\, \mathtt{in}\, e \mid \\
 & & & \mathtt{if}\, v = v\, \mathtt{then}\, e\, \mathtt{else}\, e \\
\text{Frame-stacks} & s & ::= & \mathtt{id} \mid s \circ (\lambda x\, e)
\end{array}
$$

Den sem in $\mathcal{N}dom$: $\begin{cases} V &= \mathbb{A}_\perp \oplus (V \multimap E) \\ E &= S \multimap \mathbf{1}_\perp \\ S &= V \multimap \mathbf{1}_\perp \end{cases}$

$[\![\mathtt{new}]\!] \in E = S \multimap \mathbf{1}_\perp$ maps each $f \in S = V \multimap \mathbf{1}_\perp$ to

$[\![\mathtt{new}]\!](f) \triangleq f(a)$ for some/any $a \notin supp(f)$

Example: dynamic allocation

if $a, a' \notin supp(f)$, then $(a\ a')\cdot f = f$
and so

$$f(a) = (a\ a')\cdot fa \quad (\text{since } fa \in 1_\perp = \{\top, \perp\})$$
$$= ((a\ a')\cdot f)((a\ a')\cdot a)$$
$$= f(a')$$

$[\![\mathtt{new}]\!] \in E = S \multimap \mathbf{1}_\perp$ maps each $f \in S = V \multimap \mathbf{1}_\perp$ to

$[\![\mathtt{new}]\!](f) \triangleq f(a)$ for some/any $a \notin supp(f)$

Example: dynamic allocation

Untyped Pitts-Stark ν-calculus :

Values	v	$::=$	$x \mid a \mid \lambda x\, e$
Expressions	e	$::=$	$v \mid \mathtt{new} \mid v\, v \mid \mathtt{let}\, x = e\, \mathtt{in}\, e \mid$
			$\mathtt{if}\, v = v\, \mathtt{then}\, e\, \mathtt{else}\, e$
Frame-stacks	s	$::=$	$\mathtt{id} \mid s \circ (\lambda x\, e)$

Termination relation $\langle s, e \rangle\!\downarrow$ between closed frame-stacks and closed expressions

Theorem ("computational adequacy").

$$\langle s, e \rangle\!\downarrow \;\Leftrightarrow\; [\![e]\!]([\![s]\!]) = \top$$

where $\mathbf{1}_{\perp} = \{\top, \perp\}$.

Example: dynamic allocation

Untyped Pitts-Stark ν-calculus :

$$
\begin{array}{llll}
\text{Values} & v & ::= & x \mid a \mid \lambda x\, e \\
\text{Expressions} & e & ::= & v \mid \mathtt{new} \mid v\, v \mid \mathtt{let}\ x = e\ \mathtt{in}\ e \mid \\
 & & & \mathtt{if}\ v = v\ \mathtt{then}\ e\ \mathtt{else}\ e \\
\text{Frame-stacks} & s & ::= & \mathtt{id} \mid s \circ (\lambda x\, e)
\end{array}
$$

Termination relation $\langle s, e \rangle \downarrow$ between closed frame-stacks and closed expressions

<u>Theorem</u> ("computational adequacy").

$$\langle s, e \rangle \downarrow \;\Leftrightarrow\; [\![e]\!]([\![s]\!]) = \top$$

where $\mathbf{1}_\perp = \{\top, \perp\}$.

far from fully abstract (cf. Laird), but still useful (cf Shinwell-Pitts)

Example: normalization-by-evaluation (NBE)

First have to describe the nominal domain $[\mathbb{A}]\boldsymbol{D}$ of name-bindings associated with each $\boldsymbol{D} \in \boldsymbol{\mathcal{N}dom}$...

Name-binding

$$[\mathbb{A}](-) : \mathcal{N}dom \to \mathcal{N}dom$$

Locally continuous functor given by
$[\mathbb{A}]D \triangleq (\mathbb{A} \times D)/\preceq$, where pre-order $\preceq$ is:

$$(a, d) \preceq (a', d') \triangleq (a\ a'') \cdot d \sqsubseteq (a'\ a'') \cdot d'$$

for some/any
$a'' \notin supp(a, d, a', d')$

Write $\langle a \rangle d$ for the equivalence class of (a, d).

Name-binding

$[\mathbb{A}](-) : \mathcal{N}dom \to \mathcal{N}dom$

Locally continuous functor given by
$[\mathbb{A}]D \triangleq (\mathbb{A} \times D)/\preceq$, where pre-order $\preceq$ is:

$$(a,d) \preceq (a',d') \triangleq (a\ a'')\cdot d \sqsubseteq (a'\ a'')\cdot d'$$

for some/any
$a'' \notin supp(a,d,a',d')$

Write $\langle a\rangle d$ for the equivalence class of (a,d).

fact : $\mathrm{Supp}(\langle a\rangle d) = \mathrm{supp}(d) - a$

Name-binding

$[\mathbb{A}](-) : \mathcal{N}dom \to \mathcal{N}dom$

Locally continuous functor given by
$[\mathbb{A}]D \triangleq (\mathbb{A} \times D)/\preceq$, where pre-order $\preceq$ is:

$$(a, d) \preceq (a', d') \triangleq (a\ a'') \cdot d \sqsubseteq (a'\ a'') \cdot d'$$

for some/any
$a'' \notin supp(a, d, a', d')$

Write $\langle a \rangle d$ for the equivalence class of (a, d).

quotient makes chain completeness delicate, but ...

<u>Fact</u>: any f.s. chain $e_0 \sqsubseteq e_1 \sqsubseteq e_2 \sqsubseteq \cdots$ in $[\mathbb{A}]D$ takes the form $\langle a \rangle d_0 \sqsubseteq \langle a \rangle d_1 \sqsubseteq \langle a \rangle d_2 \sqsubseteq \cdots$ for a single atom a and chain $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \cdots$ in D.

Name-binding

$[\mathbb{A}](-) : \mathcal{N}dom \to \mathcal{N}dom$

Since locally continuous, can use $[\mathbb{A}](-)$ in recursive domain equations. E.g.

$$L = \mathbb{A}_\perp \oplus (L \otimes L) \oplus ([\mathbb{A}]L)$$

Theorem. L is isomorphic to the flat nominal domain $\Lambda_\perp$, where Λ = nominal set of λ-terms (mod α).

Name-binding

$[\mathbb{A}](-) : \mathcal{N}dom \to \mathcal{N}dom$

Since locally continuous, can use $[\mathbb{A}](-)$ in recursive domain equations. E.g.

$$L = \mathbb{A}_\perp \oplus (L \otimes L) \oplus ([\mathbb{A}]L)$$

<u>Theorem</u>. L is isomorphic to the flat nominal domain $\Lambda_\perp$, where Λ = nominal set of λ-terms (mod α).

variables

Name-binding

$$[\mathbb{A}](-) : \mathcal{N}dom \to \mathcal{N}dom$$

Since locally continuous, can use $[\mathbb{A}](-)$ in recursive domain equations. E.g.

$$L = \mathbb{A}_\perp \oplus (L \otimes L) \oplus ([\mathbb{A}]L)$$

Theorem. L is isomorphic to the flat nominal domain $\Lambda_\perp$, where Λ = nominal set of λ-terms (mod α).

application terms

Name-binding

$$[\mathbb{A}](-) : \mathcal{N}dom \to \mathcal{N}dom$$

Since locally continuous, can use $[\mathbb{A}](-)$ in recursive domain equations. E.g.

$$L = \mathbb{A}_\perp \oplus (L \otimes L) \oplus ([\mathbb{A}]L)$$

Theorem. L is isomorphic to the flat nominal domain $\Lambda_\perp$, where Λ = nominal set of λ-terms (mod α).

λ-abstraction terms

Example: NBE

λ-terms	L	$=$	$\mathbb{A}_\perp \oplus (L \otimes L) \oplus ([\mathbb{A}]L)$
semantic nfs	N	$=$	$U \oplus (N \to N)$
neutrals	U	$=$	$\mathbb{A}_\perp \oplus (U \otimes N)$

normalization	$reify \circ eval$	$:$	$L \multimap L$
reification	$reify$	$:$	$N \multimap L$
evaluation	$eval$	$:$	$L \multimap N$

reify and *eval* are defined by fixpoint recursion, the interesting clause of which is:

$$reify(f \in N \to N) = \langle a \rangle (reify(f\, a))$$

for some/any $a \notin supp(f)$

Example: NBE

λ-terms	L	$=$	$\mathbb{A}_\perp \oplus (L \otimes L) \oplus ([\mathbb{A}]L)$
semantic nfs	N	$=$	$U \oplus (N \longrightarrow N)$
neutrals	U	$=$	$\mathbb{A}_\perp \oplus (U \otimes N)$

normalization	$\mathit{reify} \circ \mathit{eval}$	:	$L \multimap L$
reification	reify	:	$N \multimap L$
evaluation	eval	:	$L \multimap N$

reify and eval are defined by fixpoint recursion, the interesting clause of which is:

$$\mathit{reify}(f \in N \longrightarrow N) = \langle a \rangle (\mathit{reify}(f\, a))$$

see FreshML programming example in Fig. 7 of Shinwell, Pitts & Gabbay, Proc. ICFP'03.

for some/any

$a \notin \mathit{supp}(f)$

Topological considerations

Difficulty: name-binding construct on nominal posets, $D \mapsto [\mathbb{A}]D$

- preserves "has lubs of f.s. ω-chains", but
- does not preserve "has lubs of f.s. directed subsets"

Topological considerations

Difficulty: name-binding construct on nominal posets, $D \mapsto [\mathbb{A}]D$

- preserves "has lubs of f.s. ω-chains", but
- does not preserve "has lubs of f.s. directed subsets"

Reason: $S = \{d_n \mid n \in \mathbb{N}\}$ is in $P_{\mathrm{fs}}(D)$ iff

there is a single finite set $\overline{a}$ of atoms supporting all $d \in S$ simultaneously

but in general $S \in P_{\mathrm{fs}}(D)$ does not have this "uniformly bounded" property.

equivalently : $\boxed{\forall d \in S.\ \mathrm{supp}(d) \subseteq \mathrm{supp}(S)}$

Topological considerations

Basing nominal domain theory on

lubs of uniformly bounded directed sets

restores $D \mapsto [\mathbb{A}]D$ with good properties.

Used by

- Laird — FM-biorders model of ν-calculus
- Winskel-Turner — nominal semantics of higher-order concurrent processes with name generation

Topological considerations

Basing nominal domain theory on

lubs of uniformly bounded directed sets

restores $D \longmapsto [\mathbb{A}]D$ with good properties.

Questions:

- Does "uniformly bounded" have a characterisation within the internal HO logic of $\mathcal{N}om$?
- Is there a useful theory of nominal Scott domains / information systems / "domain theory in logical form"?

Further developments

Dynamic allocation

- N Benton and B Leperchey, “Relational Reasoning in a Nominal Semantics for Storage”, Proc. TLCA 2005 (SLNCS 3461).
- J Laird, “Sequentiality and the CPS Semantics of Fresh Names”, Proc. MFPS 23 (ENTCS 173(2007)203–219).
- N Tzevelekos, “Full abstraction for nominal general references”, Proc. LICS 2007 (building on Abramsky et al, Proc. LICS 2004).

NBE

- Sect. 6 of AMP, “Alpha-Structural Recursion and Induction”, JACM 53(2006)459–506.
- J Schwinghammer, “On Normalization by Evaluation for Object Calculi”, Proc. TYPES’07 (SLNCS 4941(2008)173–187).

Further developments

Dynamic allocation

parametric logical relations meets Ndom

- N Benton and B Leperchey, "Relational Reasoning in a Nominal Semantics for Storage", Proc. TLCA 2005 (SLNCS 3461).
- J Laird, "Sequentiality and the CPS Semantics of Fresh Names", Proc. MFPS 23 (ENTCS 173(2007)203–219).
- N Tzevelekos, "Full abstraction for nominal general references", Proc. LICS 2007 (building on Abramsky et al, Proc. LICS 2004).

game semantics in Nom with refinements ("strong support")

NBE

- Sect. 6 of AMP, "Alpha-Structural Recursion and Induction", JACM 53(2006)459–506.
- J Schwinghammer, "On Normalization by Evaluation for Object Calculi", Proc. TYPES'07 (SLNCS 4941(2008)173–187).

Future applications?

Key strength of nominal sets:

finite support generalizes "set of free variables" from syntactical data to abstract mathematical objects, such as extensional functions.

Should try to exploit this for denotational models of languages/logics that mix up syntax and semantics.

E.g. Programming languages involving reflection, staged meta-programming, ...

Suggestions welcome!