Nominal Techiques: from Nominal Logic to Nominal Games

Nikos Tzevelekos
Oxford University Computing Laboratory

LINT workshop, Amsterdam, December 2008

What this talk is about

- Nominal Techniques := formal techniques for names,
- Names := identifiers/atoms in constructions.

There are two parts in this talk; nominal techniques for:

- abstract syntax,
- semantics.

Different issues, same techniques.

From Nominal Logic to Nominal Sets

Name-binding

$$\int_0^1 f(x)\ dx$$

In the above expression we say that x *is bound* in $\int_0^1 f(x)\ dx$. Alternatively, the costructor $\int_0^1 \_\ dx$ *binds* x.

This is a very well understood notion: for example, we can easily spot the error below.

$$\int_0^1 \int_0^1 xy\ dx\,dy = \int_0^1 \int_0^1 xx\ dx\,dy = \int_0^1 \frac{1}{3}\ dx = \frac{1}{3}$$

Being formal about name-binding

Consider the simply-typed λ-calculus.

Types	$A, B ::= \ \mathrm{B} \mid A \to B$
Terms	$M, N ::= \ x \mid MN \mid \lambda x.M$

The constructor $\lambda x.\_$ is a binder. We consider terms *modulo choices of names in binding positions*. That is,

$$\mathrm{Term} := \ \mathrm{Var} + (\mathrm{Term} \times \mathrm{Term}) + (\mathrm{Var} \times \mathrm{Term})$$
$$\alpha\mathrm{Term} := \mathrm{Term}/_{=_\alpha}$$

where $M =_\alpha M'$ if M and M' differ solely in their choices of bound names.

The problem

$$\text{Term} := \text{Var} + (\text{Term} \times \text{Term}) + (\text{Var} \times \text{Term})$$
$$\alpha\text{Term} := \text{Term}/_{=_\alpha}$$

Most of the times:

- we say that we use $[M]_\alpha \in \alpha\text{Term}$,
- but in fact we use (specific!) $M' \in [M]_\alpha$.

This introduces (at best) an amount of informality in definitions and proofs regarding α-terms.

Typing rules for α-terms.

$$\frac{(x{:}A)\in\Gamma}{\Gamma\vdash x:A}\qquad\frac{\Gamma\vdash M:A\to B\quad \Gamma\vdash N:A}{\Gamma\vdash MN:B}\qquad\frac{\Gamma,x:A\vdash M:B\quad x\notin\mathsf{dom}(\Gamma)}{\Gamma\vdash\lambda x.M:A\to B}$$

What does this formally mean?

- That $\Gamma\vdash[M]_\alpha : A$ has a derivation if $\Gamma\vdash M:A$ does?

An example

Typing rules for α-terms.

$$\frac{(x{:}A)\in\Gamma}{\Gamma\vdash x:A}\qquad\frac{\Gamma\vdash M:A\to B\quad\Gamma\vdash N:A}{\Gamma\vdash MN:B}\qquad\frac{\Gamma,x:A\vdash M:B\quad x\notin\mathsf{dom}(\Gamma)}{\Gamma\vdash\lambda x.M:A\to B}$$

What does this formally mean?

- That $\Gamma\vdash[M]_\alpha:A$ has a derivation if $\Gamma\vdash M:A$ does?
- That $\Gamma\vdash[M]_\alpha:A$ has a derivation if $\Gamma\vdash M':A$ does, some $M'\in[M]_\alpha$?

Typing rules for α-terms.

$$\frac{(x{:}A)\in\Gamma}{\Gamma\vdash x:A}\qquad\frac{\Gamma\vdash M:A\to B\quad\Gamma\vdash N:A}{\Gamma\vdash MN:B}\qquad\frac{\Gamma,x:A\vdash M:B\quad x\notin\mathsf{dom}(\Gamma)}{\Gamma\vdash\lambda x.M:A\to B}$$

What does this formally mean?

- That $\Gamma \vdash [M]_\alpha : A$ has a derivation if $\Gamma \vdash M : A$ does?
- That $\Gamma \vdash [M]_\alpha : A$ has a derivation if $\Gamma \vdash M' : A$ does, some $M' \in [M]_\alpha$?
- That derivations are considered modulo α-equivalence and that $\Gamma \vdash [M]_\alpha : A$ has a derivation $[\mathcal{D}]_\alpha$ if $\Gamma \vdash M' : A$ has a derivation $\mathcal{D}$, some ("sufficiently fresh") $M' \in [M]_\alpha$?

The desideratum

- Can't we do things in a way that is both simple *and* formal?
- In particular, can't we have a syntax which directly incorporates name-binding?

$$\alpha\mathrm{Term} := \ \mathrm{Var} + (\alpha\mathrm{Term} \times \alpha\mathrm{Term}) + \langle\mathrm{Var}\rangle\alpha\mathrm{Term}$$

[Pitts, 2001]: "A first order theory of names and binding".
A many-sorted logic with:

- sorts for *data*, *names* and *name-abstractions*:

$$\mathrm{S} ::= \mathrm{A} \mid \mathrm{D} \mid \langle \mathrm{A} \rangle \mathrm{S}$$

- constructors for functions; in particular:
  - if $t_1, t_2 : \mathrm{A}$, $t : \mathrm{S}$ then $(t_1\ t_2) \cdot t : \mathrm{S}$,
  - if $t_1 : \mathrm{A}$, $t : \mathrm{S}$ then $t_1.t : \langle \mathrm{A} \rangle \mathrm{S}$,
- constructors for relations; in particular:
  - if $t_1 : \mathrm{A}$, $t : \mathrm{S}$ then $t_1 \# t$ is a formula,
- quantfiers $\forall, \exists,$ И,
- axioms.

$$\frac{t_1, t_2 : \mathrm{A} \quad t : \mathrm{S}}{(t_1\ t_2)\cdot t : \mathrm{S}}$$

$$\frac{t_1 : \mathrm{A} \quad t : \mathrm{S}}{t_1.t : \langle \mathrm{A} \rangle \mathrm{S}}$$

$$\frac{t_1 : \mathrm{A} \quad t : \mathrm{S}}{t_1 \# t : \mathsf{wff}}$$

$$\dots$$

Example axioms:

$$\text{И}a : \mathrm{A}.\ \phi(\vec{x}) \iff \exists a : \mathrm{A}.\ a\#\vec{x} \wedge \phi(\vec{x}) \qquad \text{(Q)}$$

$$\frac{t_1, t_2 : \mathrm{A} \quad t : \mathrm{S}}{(t_1\ t_2)\cdot t : \mathrm{S}}$$

$$\frac{t_1 : \mathrm{A} \quad t : \mathrm{S}}{t_1.t : \langle \mathrm{A}\rangle \mathrm{S}}$$

$$\frac{t_1 : \mathrm{A} \quad t : \mathrm{S}}{t_1 \# t : \mathsf{wff}}$$

$\ldots$

Example axioms (note sorts should match):

$$\text{И}a : \mathrm{A}.\ \phi(\vec{x}) \iff \exists a : \mathrm{A}.\ a\#\vec{x} \wedge \phi(\vec{x}) \qquad \text{(Q)}$$

$$(a\ a')\cdot(b\ b')\cdot x = ((a\ a')\cdot b\ (a\ a')\cdot b')\cdot(a\ a')\cdot x \qquad \text{(E1)}$$

$$b\#x \implies (a\ a')\cdot b\,\#\,(a\ a')\cdot x \qquad \text{(E2)}$$

$$a\#x \wedge a'\#x \implies (a\ a')\cdot x = x \qquad \text{(F1)}$$

$$a.x = a'.x' \iff (a = a' \vee a'\#x) \wedge x' = (a\ a')\cdot x \qquad \text{(A1)}$$

NL gives us a strong handle on names. For example:

- $\phi(\vec{x}) \iff \phi((a\ a')\cdot\vec{x})$
- $(\exists a : \mathrm{A}.\ a\#\vec{x} \wedge \phi(\vec{x})) \iff (\forall a : \mathrm{A}.\ a\#\vec{x} \implies \phi(\vec{x}))$
- $b\,\#\,a.x \iff b = a \vee b\,\#\,x$
- $a.x = a'.x' \iff \text{И}b : \mathrm{A}.\ (a\ b)\cdot x = (a'\ b)\cdot x'$

$$\frac{t_1, t_2 : \mathrm{A} \quad t : \mathrm{S}}{(t_1\ t_2) \cdot t : \mathrm{S}}$$

$$\frac{t_1 : \mathrm{A} \quad t : \mathrm{S}}{t_1.t : \langle \mathrm{A} \rangle \mathrm{S}}$$

$$\frac{t_1 : \mathrm{A} \quad t : \mathrm{S}}{t_1 \# t : \mathsf{wff}}$$

$\dots$

Consider a countably infinite set $\mathbb{A}$ of *atoms* and its group of finite permutations PERM($\mathbb{A}$).

A *nominal set* is a pair $(X, \cdot)$ such that X is a set and

- $\_ \cdot \_ :$ PERM$(\mathbb{A}) \times X \to X$ is an action on X,
  - i.e. $\mathsf{id} \cdot x = x, \quad \pi \cdot (\pi' \cdot x) = (\pi \circ \pi') \cdot x$,
- each $x \in X$ has *finite support*,
  - i.e, there exists finite $S \subseteq \mathbb{A}$,
    $\forall \pi.\ (\forall a \in S.\ \pi(a) = a) \implies \pi \cdot x = x$,

In particular, each $x \in X$ has a *least support*, $\mathrm{supp}(x)$.

$$\frac{t_1, t_2 : \mathrm{A} \quad t : \mathrm{S}}{(t_1\ t_2) \cdot t : \mathrm{S}}$$

$$\frac{t_1 : \mathrm{A} \quad t : \mathrm{S}}{t_1.t : \langle \mathrm{A} \rangle \mathrm{S}}$$

$$\frac{t_1 : \mathrm{A} \quad t : \mathrm{S}}{t_1 \# t : \mathsf{wff}}$$

$\ldots$

Consider a countably infinite set $\mathbb{A}$ of *atoms* and its group of finite permutations PERM($\mathbb{A}$).

A *nominal set* is a pair $(X, \cdot)$ such that X is a set and

- $\_ \cdot \_ : \mathrm{PERM}(\mathbb{A}) \times X \to X$ is an action on X,
  - i.e. $\mathrm{id} \cdot x = x, \quad \pi \cdot (\pi' \cdot x) = (\pi \circ \pi') \cdot x$,
- each $x \in X$ has *finite support*,
  - i.e, there exists finite $S \subseteq \mathbb{A}$,
    $\forall \pi.\ (\forall a \in S.\ \pi(a) = a) \implies \pi \cdot x = x$,

In particular, each $x \in X$ has a *least support*, $\mathrm{supp}(x)$.
For example, any set is trivially nominal, $\mathbb{A}$ is a nominal set, products of nominal sets are nominal, etc.

$\sim$ Nominal sets derived from FM permutation models of ZFA.

$$\frac{t_1, t_2 : \mathrm{A} \quad t : \mathrm{S}}{(t_1\ t_2) \cdot t : \mathrm{S}}$$

$$\frac{t_1 : \mathrm{A} \quad t : \mathrm{S}}{t_1.t : \langle \mathrm{A} \rangle \mathrm{S}}$$

$$\frac{t_1 : \mathrm{A} \quad t : \mathrm{S}}{t_1 \# t : \textsf{wff}}$$

$\ldots$

Nominal sets provide a model for NL:

- map each D to some X_{D},
- map A to $\mathbb{A}$,
- for each $a, b \in \mathbb{A}$ and $x \in X$ take:
  - $(a\ b) \cdot x$ as given,
  - $a \# x$ if $a \notin \mathrm{supp}(x)$,
  - $a.x := \{(b, y) \mid (a = b \vee b \# y) \wedge y = (a\ b) \cdot x\}$.

  Thus, $\langle \mathbb{A} \rangle X := \{a.x \mid a \in \mathbb{A} \wedge x \in X\}$.
- etc.

Remarks before continuing

- Nominal techniques introduced in [Gabbay & Pitts'99]. The original presentation was set-theoretic, in ZFA.
- Nominal techniques have had a huge impact on abstract syntax:
  - nominal algebras,
  - nominal rewriting systems,
  - nominal theorem provers,
  - nominal metalanguages, etc.

  See e.g. works of: Cheney, Gabbay, Mathijssen, Pitts, Shinwell, Urban, and collaborators.
- but also in semantics via nominal sets:
  - nominal domains [Shinwell & Pitts],
  - nominal games.

From Nominal Sets to Nominal Games

Denotational Semantics assigns to terms denotations in some abstract mathematical domain (a category).

- Issues with α-equivalence disappear at the level of semantics.
- Different approach: $\lambda x.f(x)$ represents
  - a name-abstraction (no comp. content) in syntax,
  - an exponential (a function) in semantics.

Denotational Semantics assigns to terms denotations in some abstract mathematical domain (a category).

- Issues with α-equivalence disappear at the level of semantics.
- Different approach: $\lambda x.f(x)$ represents
  - a name-abstraction (no comp. content) in syntax,
  - an exponential (a function) in semantics.
- But there is still space for nominal techniques, in *languages with names*:
  - names for references,
  - names for objects, exceptions,
  - names for threads, channels, etc.

Types	$A, B ::=$	$\mathbf{B} \mid \mathbf{A} \mid A \to B$	base types: booleans, names
Terms	$M, N ::=$	$\mathrm{t} \mid \mathrm{f} \mid x \mid MN \mid \lambda x.M$	λ-calculus over booleans
		$\mid$ if M then N_1 else N_2	
		$\mid a$	name
		$\mid [M_1 = M_2]$	name-equality test
		$\mid \nu a.M$	name-abstraction

Types	$A, B ::=$	$\mathbf{B} \mid \mathbf{A} \mid A \to B$	base types: booleans, names
Terms	$M, N ::=$	$\mathrm{t} \mid \mathrm{f} \mid x \mid MN \mid \lambda x.M$	λ-calculus over booleans
		$\mid$ if M then N_1 else N_2	
		$\mid a$	name
		$\mid [M_1 = M_2]$	name-equality test
		$\mid \nu a.M$	name-abstraction

- Terms form a nominal set ($a \in \mathbb{A}$).
- $\nu a.M$ creates a *fresh* name a for M – it is a binder.
- Terms are taken modulo α-equivalence (wrt both bindings).

Types	$A, B ::=$	$\mathbf{B} \mid \mathbf{A} \mid A \to B$	base types: booleans, names
Terms	$M, N ::=$	$\mathrm{t} \mid \mathrm{f} \mid x \mid MN \mid \lambda x.M$	λ-calculus over booleans
		$\mid$ if M then N_1 else N_2	
		$\mid a$	name
		$\mid [M_1 = M_2]$	name-equality test
		$\mid \nu a.M$	name-abstraction

- Terms form a nominal set ($a \in \mathbb{A}$).
- $\nu a.M$ creates a *fresh* name a for M – it is a binder.
- Terms are taken modulo α-equivalence (wrt both bindings).

"Names are created with local scope, can be tested for equality and can be passed around via function application, but that is all."

- Reduction happens in *state-environments*. Reduction rules include:
  - λ-calculus rules (call by value),
  - nominal rules:

$$S,\ \nu a.M \to S \oplus a,\ M$$
$$S,\ [a = b] \to S,\ \mathrm{f} \quad (a \neq b)$$
$$S,\ [a = a] \to S,\ \mathrm{t}$$

  So reduction is non-deterministic, in a "nominal way".

- Two terms are *(observationally) equivalent* ($\cong$) if no context of type $\mathrm{C}[\_] : \mathbf{B}$ can distinguish them.

This simple calculus is quite expressive. For example:

- $$\nu a.\lambda x.a \ncong \lambda x.\nu a.a$$
  $$\nu a.\lambda x.[a = x] \cong \lambda x.\mathrm{f}$$
  $$\ldots$$
- As n ranges in ω we get infinitely many (observationally) different terms of type $\mathbf{A} \to \mathbf{A}$ by:

$$\begin{aligned}\nu a_1.\ldots\nu a_n.\lambda x.\ & \text{if } [x = a_1] \text{ then } a_2 \text{ else} \\ & \text{if } [x = a_2] \text{ then } a_3 \text{ else} \\ & \ldots \\ & \text{if } [x = a_{n-1}] \text{ then } a_n \text{ else } a_1\end{aligned}$$

Although introduced in [Pitts & Stark, 1993], its first fully abstract semantics was given in [AGMOS, 2004].

[AGMOS'04] and [Laird'04] introduced Nominal Games.

Names excluded, the ν-calculus is game-semantically easy.
The extra feature needed was *plays-with-names*:

- names in plays as first-class moves (like integers),
- strategies unable to distinguish between fresh names (unlike integers),
- some notion of *local state* (or *name-availability*).

[AGMOS'04] and [Laird'04] introduced Nominal Games.

Names excluded, the ν-calculus is game-semantically easy. The extra feature needed was *plays-with-names*:

- names in plays as first-class moves (like integers),
- strategies unable to distinguish between fresh names (unlike integers),
- some notion of *local state* (or *name-availability*).

All of the above achieved elegantly by use of nominal sets at the basis of moves, plays, strategies, etc.

This is no coincidence: the first two specifications go back to the notions of atomic, bindable names at the very basis of nominal techniques!

Conclusions in Nominal Games

[What has been accomplished] A series of FA models:

- for the ν-calculus [AGMOS'04, Tz'07],
- ν-calc.+HO-references,exceptions [Tz'07, Tz'08],
- ν-calc.+pointers [Laird'04, Laird'08],
- ν-calc.+HO-concurrency [Laird'06],
- ν-calc.+int-references [Tz & Murawski'08].

[What to do next] Examine (at least):

- more nominal languages (...),
- decidability of nominal languages,
- other structures under the "nominal lense" (e.g. AJM-games)!

Conclusions in Nominal Games

[What has been accomplished] A series of FA models:

- for the ν-calculus [AGMOS'04, Tz'07],
- ν-calc.+HO-references,exceptions [Tz'07, Tz'08],
- ν-calc.+pointers [Laird'04, Laird'08],
- ν-calc.+HO-concurrency [Laird'06],
- ν-calc.+int-references [Tz & Murawski'08].

[What to do next] Examine (at least):

- more nominal languages (...),
- decidability of nominal languages,
- other structures under the "nominal lense" (e.g. AJM-games)!

THANKS!