Nominal Logic: A First Order Theory of Names and Binding

Andrew M. Pitts

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Explicitly named bound variables

Aim to give a formal logic for some informal practices when representing and reasoning with syntax involving explicitly named bound variables.

Explicitly named bound variables

Aim to give a formal logic for some informal practices when representing and reasoning with syntax involving explicitly named bound variables.

Such as: "*. . . by induction on the structure of parse trees, but renaming bound variables to be fresh as necessary*" (cf. Barendregt Variable Convention) —isn't a correct use of structural induction.

Explicitly named bound variables

Don't aim to replace explicitly named bound variables with *anonymous* forms of binder.

- Use of de Bruijn indices doesn't address common, informal practices.
- Use of meta-level typed lambda calculus (HOAS) has problems with *simple* forms of structural recursion/induction.

Explicitly named bound variables

Previous work (joint with MJ Gabbay): mathematical model of *syntax modulo* α*-equivalence* with good structural recursion/induction properties. Uses *Fraenkel-Mostowski permutation model of set theory* (FM-sets).

Explicitly named bound variables

Previous work (joint with MJ Gabbay): mathematical model of *syntax modulo α-equivalence* with good structural recursion/induction properties. Uses *Fraenkel-Mostowski permutation model of set theory* (FM-sets).

This work: *Nominal logic* gives a simple, first-order axiomatisation of key properties of the FM-sets model.

Fundamental assumption underlying Nominal Logic

The only assertions [about syntax] *we deal with are* **equivariant**, *i.e. their validity is invariant under swapping bindable names.*

Fundamental assumption underlying Nominal Logic

The only assertions [about syntax] *we deal with are* **equivariant**, *i.e. their validity is invariant under swapping bindable names.*

Swapping $(a\,a')\cdot t$, not *renaming* $[a'/a]t$, because swapping-invariant properties have better logical properties than renaming-invariant ones, while sufficing for a theory of syntax mod α-equivalence

Fundamental assumption underlying Nominal Logic

The only assertions [about syntax] *we deal with are* **equivariant**, *i.e. their validity is invariant under swapping* *bindable* *names.*

Bindable names rather than *bound* names because, for reasons of compositionality, we have to deal with "bits of syntax"

Fundamental assumption underlying Nominal Logic

The only assertions [about syntax] *we deal with are* **equivariant**, *i.e. their validity is invariant under swapping bindable names.*

Bindable names are called **atoms** in Nominal Logic—mathematically different from names of *constants* (the latter are not subject to swapping)

Nominal Logic

is many-sorted first-order logic with equality ($\neg$, $\wedge$, $\vee$, $\exists$, $\forall$, $=$) plus:

- some sorts are designated **sorts of atoms**
- terms with explicit **swapping** of atoms
- **freshness** relation and quantifier
- **atom-abstraction** sorts and terms

Nominal Logic

is many-sorted first-order logic with equality ($\neg$, $\wedge$, $\vee$, $\exists$, $\forall$, $=$) plus:

- some sorts are designated **sorts of atoms**
- terms with explicit **swapping** of atoms
- **freshness** relation and quantifier
- **atom-abstraction** sorts and terms

Explicit swapping

Given terms $a : A$, $a' : A$ and $t : S$, with A a sort of atoms and S any sort, there is a term

$$(a\,a')\cdot t$$

of sort S, read "*swap a and a' in t*".

Explicit swapping

Axioms

$$(a\,a)\cdot x = x$$

Explicit swapping

Axioms

$$(a\,a)\cdot x = x$$

$$(a\,a')\cdot(a\,a')\cdot x = x$$

Explicit swapping

Axioms

$$(a\,a)\cdot x = x$$

$$(a\,a')\cdot(a\,a')\cdot x = x$$

$$\begin{array}{ll} & (a_1\,a_2)\cdot a_3 = a_3' \\ \wedge & (a_1\,a_2)\cdot a_4 = a_4' \\ \Rightarrow & (a_1\,a_2)\cdot(a_3\,a_4)\cdot x = \\ & \qquad (a_3'\,a_4')\cdot(a_1\,a_2)\cdot x \end{array}$$

Explicit swapping

Theorem. *In any model, the transposition action* $x \mapsto (a\,a')\cdot x$ *extends uniquely to a* **permutation action** *of all finite, sort-respecting permutations of atoms.*

Explicit swapping

Theorem. *In any model, the transposition action* $x \mapsto (a\,a')\cdot x$ *extends uniquely to a* **permutation action** *of all finite, sort-respecting permutations of atoms.*

$id\cdot x = x$,
$\pi\cdot(\pi'\cdot x) = (\pi\pi')\cdot x$

Explicit swapping

Theorem. *In any model, the transposition action* $x \mapsto (a\,a')\cdot x$ *extends uniquely to a* **permutation action** *of all finite, sort-respecting permutations of atoms.*

Proof uses one of the standard presentations of the symmetric group on finitely many symbols.

Equivariance

Axioms

$$(a\,a')\cdot f(x_1,\ldots,x_n) = f((a\,a')\cdot x_1,\ldots,(a\,a')\cdot x_n)$$

(each function symbol $f : S_1,\ldots,S_n \to S$)

Equivariance

Axioms

$$(a\,a')\cdot f(x_1,\ldots,x_n) = f((a\,a')\cdot x_1,\ldots,(a\,a')\cdot x_n)$$

(each function symbol $f : S_1,\ldots,S_n \to S$)

(Note that axiom

$$\begin{array}{ll} & (a_1\,a_2)\cdot a_3 = a_3' \\ \wedge & (a_1\,a_2)\cdot a_4 = a_4' \\ \Rightarrow & (a_1\,a_2)\cdot(a_3\,a_4)\cdot x = \\ & \quad (a_3'\,a_4')\cdot(a_1\,a_2)\cdot x \end{array}$$

says the swapping functions are equivariant.)

Equivariance

Axioms

$$(a\,a')\cdot f(x_1,\ldots,x_n) = \\ f((a\,a')\cdot x_1,\ldots,(a\,a')\cdot x_n)$$

(each function symbol $f : S_1,\ldots,S_n \to S$)

$$R(x_1,\ldots,x_n) \Leftrightarrow \\ R((a\,a')\cdot x_1,\ldots,(a\,a')\cdot x_n)$$

(each relation symbol $R <: S_1,\ldots,S_n$)

Equivariance

Theorem. *Each first-order formula* $\varphi(x_1, \ldots, x_n)$ (with free variables among those indicated) *satisfies the equivariance property*:

$$\varphi(x_1, \ldots, x_n) \Leftrightarrow$$
$$\varphi((a\,a')\cdot x_1, \ldots, (a\,a')\cdot x_n)$$

Nominal Logic

is many-sorted first-order logic with equality ($\neg$, $\wedge$, $\vee$, $\exists$, $\forall$, $=$) plus:

- some sorts are designated **sorts of atoms**
- terms with explicit **swapping** of atoms
- **freshness** relation and quantifier
- **atom-abstraction** sorts and terms

Freshness relation

Given terms $a : A$ and $t : S$,
with A a sort of atoms and S any sort,
there is an atomic formula

$$a \# t$$

read "*a is fresh for t*".

Freshness relation

Axioms

equivariance property for $\#$

Freshness relation

Axioms

equivariance property for #

$$a \# x \wedge a' \# x \Rightarrow (a\,a') \cdot x = x$$

Freshness relation

Axioms

equivariance property for #

$$a \# x \wedge a' \# x \Rightarrow (a\, a') \cdot x = x$$

$$(\forall x_1 : S_1) \cdots (\forall x_n : S_n)$$
$$(\exists a : A)\, a \# x_1 \wedge \cdots \wedge a \# x_n$$

(for all sorts $S_1, \ldots, S_n$ and all sorts of atoms A)

Freshness relation

Axioms

equivariance property for $\#$

$$a \mathbin{\#} x \wedge a' \mathbin{\#} x \Rightarrow (a\,a')\cdot x = x$$

$$(\forall \vec{x} : \vec{S})(\exists a : A)\, a \mathbin{\#} \vec{x}$$

(for all sorts $\vec{S}$ and all sorts of atoms A)

Freshness quantifier

Theorem *Each first-order formula* $\varphi(a, \vec{x})$ (with free variables among those indicated) *satisfies*:

$$(\exists a : A)\, a \,\#\, \vec{x} \wedge \varphi(a, \vec{x})$$
$$\Leftrightarrow\ (\forall a : A)\, a \,\#\, \vec{x} \Rightarrow \varphi(a, \vec{x})$$

Freshness quantifier

Theorem *Each first-order formula* $\varphi(a, \vec{x})$ (with free variables among those indicated) *satisfies*:

$$(\exists a : A)\ a \,\#\, \vec{x} \wedge \varphi(a, \vec{x})$$

$$\Leftrightarrow (\forall a : A)\ a \,\#\, \vec{x} \Rightarrow \varphi(a, \vec{x})$$

Define $(\text{И}a : A)\varphi$ to be either formula (where $\vec{x} = FV(\varphi) - \{a\}$) and read it as "*for some/any fresh a, φ*"

Nominal Logic

is many-sorted first-order logic with equality ($\neg$, $\wedge$, $\vee$, $\exists$, $\forall$, $=$) plus:

- some sorts are designated **sorts of atoms**
- terms with explicit **swapping** of atoms
- **freshness** relation and quantifier
- **atom-abstraction** sorts and terms

Atom-abstraction

Sort formation: for every sort of atoms $\boldsymbol{A}$ and every sort $\boldsymbol{S}$, there is a sort

$$[\boldsymbol{A}]\boldsymbol{S}$$ "*sort of atom-abstractions*"

Atom-abstraction

Sort formation: for every sort of atoms A and every sort S, there is a sort

$$[A]S \quad \text{“}\textit{sort of atom-abstractions}\text{”}$$

Term formation: given terms $a : A$ and $t : S$, there is a term

$$a.t \quad \text{“}\textit{abstract } a \textit{ in } t\text{”}$$

of sort $[A]S$.

Atom-abstraction

Axioms

equivariance property for $a, x \longmapsto a.x$

Atom-abstraction

Axioms

equivariance property for $a, x \mapsto a.x$

extensionality

$$a.x = a'.x' \Leftrightarrow (\text{И}a'' : A)\,(a''\,a)\cdot x = (a''\,a')\cdot x'$$

Atom-abstraction

Axioms

equivariance property for $a, x \mapsto a.x$

extensionality

$$a.x = a'.x' \Leftrightarrow (\text{И}a'' : A)\,(a''\,a)\cdot x = (a''\,a')\cdot x'$$

exhaustion

$$(\forall y : [A]S)\,(\exists a : A)(\exists x : S)\; y = a.x$$

Nominal Logic

is many-sorted first-order logic with equality ($\neg$, $\wedge$, $\vee$, $\exists$, $\forall$, $=$) plus:

- some sorts are designated **sorts of atoms**
- terms with explicit **swapping** of atoms
- **freshness** relation and quantifier
- **atom-abstraction** sorts and terms

Swapping and freshness for atoms

Axioms

$$(a\,a')\cdot a = a'$$

Swapping and freshness for atoms

Axioms

$$(a\,a')\cdot a = a'$$

$$(\forall a, a' : A)\; a \,\#\, a' \Leftrightarrow \neg(a = a')$$

(A any sort of atoms)

Swapping and freshness for atoms

Axioms

$$(a\,a')\cdot a = a'$$

$$(\forall a, a' : A)\; a \# a' \Leftrightarrow \neg(a = a')$$

(A any sort of atoms)

$$(\forall a : A)(\forall a' : A')\; a \# a'$$

(A, A' any *distinct* sorts of atoms)

Summary of the axioms

Many-sorted first-order logic with equality ($\neg$, $\wedge$, $\vee$, $\exists$, $\forall$, $=$) plus axioms for

- elementary properties of swapping
- ensuring all terms and formulas are equivariant
- properties of freshness
- characterising atom-abstraction sorts up to bijection

Standard interpretation

- **Sorts** denote **FM-sets** = sets equipped with atom-permutation action for which every element is **finitely supported**

  for each element x there is a *finite* set of atoms w such that $(a\,a')\cdot x = x$ for all $a, a' \in w$

Standard interpretation

- **Sorts** denote **FM-sets** = sets equipped with atom-permutation action for which every element is finitely supported
- A **sort of atoms** denotes the set of all atoms of a particular kind, with canonical permutation action (given by application).

Standard interpretation

- **Function and relation symbols** denote equivariant functions and relations (i.e. ones preserving the permutation action).
- **Swapping:** $(a\, a')\cdot x$ = special case of the given permutation action $\pi\cdot x$, for π = the permutation interchanging a and a'.

Standard interpretation

- **Freshness relation:** $a \# x$ means "*a is not in the support of x*".
- **Freshness quantifier:** $(\text{И}a : A)\varphi(a)$ means "*$\varphi(a)$ holds for all but finitely many atoms a*".
- **Atom-abstraction sorts** $[A]S$ denote a quotient $(A \times S)/\sim$, where $\sim$ as in the extensionality axiom.

Standard interpretation

Soundness Theorem. *The standard interpretation in FM-sets satisfies all the axioms of Nominal Logic.*

Standard interpretation

Soundness Theorem. *The standard interpretation in FM-sets satisfies all the axioms of Nominal Logic.*

Incompleteness: the standard interpretation of the freshness relation uses the weak-second-order notion of *finite support*—so we should not expect Nominal Logic to be complete for standard models. For example. . .

Incompleteness example

Consider the Nominal theory

sort of atoms A; sorts N, D; function symbols $o : N$, $s : N \to N$, $f : N, D \to A$;
axioms

$$(\forall x : N)\neg(o = s(x))$$

$$(\forall x, x' : N) s(x) = s(x') \Rightarrow x = x'$$

Incompleteness example

Consider the Nominal theory

sort of atoms $\boldsymbol{A}$; sorts $\boldsymbol{N}$, $\boldsymbol{D}$; function symbols $\boldsymbol{o : N}$, $\boldsymbol{s : N \to N}$, $\boldsymbol{f : N, D \to A}$;
axioms

$$(\forall x : N)\neg(o = s(x))$$

$$(\forall x, x' : N)s(x) = s(x') \Rightarrow x = x'$$

Any standard model of it satisfies

$$(\forall y : D)(\exists x, x' : N)\,\neg(x = x') \wedge f(x, y) = f(x', y)$$

Incompleteness example

Consider the Nominal theory

sort of atoms A; sorts N, D; function symbols $o : N$, $s : N \to N$, $f : N, D \to A$;
axioms

$$(\forall x : N)\neg(o = s(x))$$

$$(\forall x, x' : N)s(x) = s(x') \Rightarrow x = x'$$

Any standard model of it satisfies

$$(\forall y : D)(\exists x, x' : N)\,\neg(x = x') \wedge f(x, y) = f(x', y)$$

but this sentence cannot be proved from the theory in Nominal Logic.

Nominal theory of λ-terms modulo α-equivalence

Intended model

$$\left(\begin{array}{c} \boldsymbol{\lambda}\text{-terms over a countable} \\ \text{set of variables } \boldsymbol{a \in A} \\ \boldsymbol{t ::= a \mid t\,t \mid \lambda a.t} \end{array} \right) \Big/ \ \boldsymbol{\alpha}\text{-equivalence}$$

isomorphic to the inductively defined FM-set

$$\boldsymbol{\mu\Lambda.(A + (\Lambda \times \Lambda) + [A]\Lambda)}$$

Nominal theory of λ-terms modulo α-equivalence

Signature

sort of atoms	A
sort	Λ
function symbols	$var : A \longrightarrow \Lambda$
	$app : \Lambda, \Lambda \longrightarrow \Lambda$
	$lam : [A]\Lambda \longrightarrow \Lambda$
relation symbol	$sub <: \Lambda, A, \Lambda, \Lambda$

Nominal theory of λ-terms modulo α-equivalence

Signature

sort of atoms	$\boldsymbol{A}$ ← *“variables”*
sort	$\boldsymbol{\Lambda}$
function symbols	$\boldsymbol{var : A \longrightarrow \Lambda}$
	$\boldsymbol{app : \Lambda, \Lambda \longrightarrow \Lambda}$
	$\boldsymbol{lam : [A]\Lambda \longrightarrow \Lambda}$
relation symbol	$\boldsymbol{sub <: \Lambda, A, \Lambda, \Lambda}$

Nominal theory of λ-terms modulo α-equivalence

Signature

sort of atoms	$\boldsymbol{A}$
sort	$\boldsymbol{\Lambda}$ ⟵ "**λ**-*terms mod* $=_\alpha$"
function symbols	$\boldsymbol{var : A \longrightarrow \Lambda}$
	$\boldsymbol{app : \Lambda, \Lambda \longrightarrow \Lambda}$
	$\boldsymbol{lam : [A]\Lambda \longrightarrow \Lambda}$
relation symbol	$\boldsymbol{sub <: \Lambda, A, \Lambda, \Lambda}$

Nominal theory of λ-terms modulo α-equivalence

Signature

sort of atoms $\boldsymbol{A}$

sort $\boldsymbol{\Lambda}$

$\boldsymbol{: A \longrightarrow \Lambda}$

$\boldsymbol{app : \Lambda, \Lambda \longrightarrow \Lambda}$

$\boldsymbol{lam : [A]\Lambda \longrightarrow \Lambda}$

relation symbol $\boldsymbol{sub <: \Lambda, A, \Lambda, \Lambda}$

$\boldsymbol{sub(t, a, t', t'')}$ supposed to mean $\boldsymbol{[t/a]t' =_\alpha t''}$

Nominal theory of λ-terms modulo α-equivalence

Axioms

"*$\boldsymbol{var}$, $\boldsymbol{app}$ and $\boldsymbol{lam}$ are injective and have disjoint images whose union is the whole of $\boldsymbol{\Lambda}$*"

Nominal theory of λ-terms modulo α-equivalence

Axioms

"var, app and lam *are injective and have disjoint images whose union is the whole of* Λ"

induction axiom...

clauses defining substitution...

Induction axiom

$$(\forall a : \mathbb{A})\,\varphi(var(a), \vec{y})$$
$$\wedge$$
$$(\forall x, x' : \Lambda)$$
$$\varphi(x, \vec{y}) \wedge \varphi(x', \vec{y}) \Rightarrow \varphi(app(x, x'), \vec{y})$$
$$\wedge$$
$$(\text{И}a : \mathbb{A})(\forall x : \Lambda)$$
$$\varphi(x, \vec{y}) \Rightarrow \varphi(lam(a.x), \vec{y})$$
$$\Rightarrow$$
$$(\forall x : \Lambda)\,\varphi(x, \vec{y})$$

Induction axiom

$$(\forall a : A)\, \varphi(var(a), \vec{u})$$

$$\wedge$$

$$(\forall x, x' : \Lambda)$$

$$\varphi(x, \vec{y}) \wedge \varphi(x$$

$$\wedge$$

$$(\text{И} a : A)(\forall x : \Lambda)$$

$$\varphi(x, \vec{y}) \Rightarrow \varphi(lam(a.x), \vec{y})$$

$$\Rightarrow$$

$$(\forall x : \Lambda)\, \varphi(x, \vec{y})$$

makes sense of "*. . . by induction on the structure of parse trees, but renaming bound variables to be fresh as necessary*"

Substitution axioms

$$sub(x, a, var(a), x)$$

Substitution axioms

$$sub(x, a, var(a), x)$$

$$a \mathrel{\#} a' \Rightarrow sub(x, a, var(a'), var(a'))$$

Substitution axioms

$$sub(x, a, var(a), x)$$

$$a \mathrel{\#} a' \Rightarrow sub(x, a, var(a'), var(a'))$$

$$sub(x, a, y, z) \wedge sub(x, a, y', z') \\ \Rightarrow sub(x, a, app(y, y'), app(z, z'))$$

Substitution axioms

$$sub(x, a, var(a), x)$$

$$a \# a' \Rightarrow sub(x, a, var(a'), var(a'))$$

$$sub(x, a, y, z) \wedge sub(x, a, y', z') \Rightarrow sub(x, a, app(y, y'), app(z, z'))$$

$$sub(x, a, y, z) \wedge a' \# x \Rightarrow sub(x, a, lam(a'.y), lam(a'.z))$$

Substitution axioms

$$sub(x, a, var(a), x)$$

$$a \# a' \Rightarrow sub(x, a, var(a'), var(a'))$$

$$sub(x, a, y, z) \wedge sub(x, a, y', z')$$
$$\Rightarrow sub(x, a, app(y, y$$

$$sub(x, a, y, z) \wedge a' \# x$$
$$\Rightarrow sub(x, a, lam(a'.y), lam(a'.z))$$

in the standard model, this means "*a' is not free in x*"

Nominal theory of λ-terms modulo α-equivalence

Sample theorem of this theory whose proof makes uses of the induction axiom:

$$(\forall x : \mathbf{\Lambda})(\forall a : \mathbb{A})(\forall y : \mathbf{\Lambda})$$
$$(\exists ! z : \mathbf{\Lambda})\ sub(x, a, y, z)$$

Conclusions

Nominal logic is an *first-order* presentation of the key concepts of the FM-sets model of syntax-with-binders:
equivariance, **freshness** and **abstraction**.

Conclusions

Nominal logic is an *first-order* presentation of the key concepts of the FM-sets model of syntax-with-binders:
equivariance, **freshness** and **abstraction**.

Being first order, it doesn't give a *complete* axiomatisation of FM-sets notion of *finite support*, but properties of freshness in Nominal Logic seem sufficient in practice (cf. Gabbay's development of an Isabelle package for FM-set theory).

Does the world really need yet another logic?!

Even if you don't buy FM-sets, Nominal Logic, etc, take home two simple but important underlying ideas, useful for operational semantics (whether pencil-and-paper or mechanised):

- Name-swapping has much nicer logical properties than renaming.
- The only assertions about syntax we should deal with are ones whose validity is invariant under swapping bindable names.