Nominal Algebra

Murdoch J. Gabbay

Università di Pisa, Dipartimento di Informatica
Monday, 30 June 2008

Thanks to Vincenzo Ciancia and Ugo Montanari

Nominal algebra

Algebra is a simple and expressive description and specification language based on equality.

It cannot easily specify basic systems of computer science such as the lambda-calculus, first-order logic, or the pi-calculus.

This is because the natural specifications of such systems involve object-level variables, meta-level variables, and freshness conditions. For example:

Nominal algebra

- λ-calculus: $(\lambda x.r)[y{\mapsto}t] = \lambda x.(r[y{\mapsto}t])$ if x is fresh for t
- λ-calculus: $\lambda x.(rx) = r$ if x is fresh for r
- π-calculus: $\nu x.(P \mid Q) = P \mid \nu x.Q$ if x is fresh for P
- First-order logic: $\forall x.(\phi \Rightarrow \psi) = \phi \Rightarrow \forall x.\psi$ if x is fresh for ϕ

These informal statements mention two levels of variable;
object-variables x, y and meta-variables r, t, P, Q, ϕ, ψ.

Capture-avoidance conditions are (freshness) constraints on the values that meta-variables may assume.

Nominal algebra

Meta-variables are naturally substituted with capturing substitution.

Consider the following quote:

“Set r to x and t to x in $(\lambda x.r)[y{\mapsto}t]$;
obtain $(\lambda x.x)[y{\mapsto}x]$.”

Nominal algebra

Capturing substitution on syntax is easy to define yet syntax has semantics, which motivates the study of non-trivial equalities.

This motivates an ‘algebraic system’ with two levels of variable, and freshness side-conditions; the result I obtained was nominal algebra.

Nominal algebra is more elementary than the λ-calculus and first/higher-order logic.

It is a language within which these can be specified — i.e. I see this as a foundational mathematical logic for foundational mathematical logics/calculi — and perhaps within which new theorems can be conveniently proved about them.

Nominal algebra

Nominal algebra uses nominal terms.

Nominal terms feature a two-level hierarchy of variables reflecting the hierarchy noted above:

- level 1 variables $a, b, c, d, \ldots$ (atoms) model object-variables;
- level 2 variables $X, Y, Z, \ldots$ (unknowns) model meta-variables.

$a\#X$ means intuitively ‘a is fresh for (the denotation of) X’.

Nominal terms

Nominal terms are inductively defined by

$$t ::= a \mid \pi X \mid [a]t \mid \mathsf{f}(t, \ldots, t).$$

Here:

- $a, b, c, \ldots$ are atoms.
- $X, Y, Z, \ldots$ are unknowns.
- $\mathsf{f}, \mathsf{g}, \ldots$ are term-formers.
- $[a]t$ is an abstraction.
- π is a permutation (finitely-supported bijection of atoms).

Theory of substitution SUB

Assume a binary term-former sub. Sugar $\mathsf{sub}([a]s, t)$ to $s[a\mapsto t]$.

$$
\begin{array}{rlrcl}
(\mathbf{var}\mapsto) & & a[a\mapsto T] & = & T \\
(\#\mapsto) & a\#X \vdash & X[a\mapsto T] & = & X \\
(\mathsf{f}\mapsto) & & \mathsf{f}(X_1,\ldots)[a\mapsto T] & = & \mathsf{f}(X_1[a\mapsto t],\ldots) \\
(\mathbf{abs}\mapsto) & b\#T \vdash & ([b]X)[a\mapsto T] & = & [b](X[a\mapsto T]) \\
(\mathbf{ren}\mapsto) & b\#X \vdash & X[a\mapsto b] & = & (b\ a)\cdot X
\end{array}
$$

Nominal algebra axioms are equalities subject to freshness side-conditions.

λ-calculus theory LAM

Assume term-formers λ, and app. Sugar app(t, u) to tu.

$$(\beta) \qquad (\lambda[a]Y)X = Y[a\mapsto X]$$

The 'a' in axioms should be interpreted equivariantly, i.e. up to permutation ('a and b' represents 'any two distinct atoms'); the 'X' should be interpreted up to substitution ('X' represents 'any term').

λ-calculus theory LAM, unpacked

$$
\begin{array}{rrrcl}
(\mathbf{var}{\mapsto}) & \vdash & a[a{\mapsto}X] & = & X \\
(\#{\mapsto}) & a\#Z \vdash & Z[a{\mapsto}X] & = & Z \\
(\mathbf{app}{\mapsto}) & \vdash & (Z'Z)[a{\mapsto}X] & = & (Z'[a{\mapsto}X])(Z[a{\mapsto}X]) \\
(\mathbf{abs}{\mapsto}) & b\#X \vdash & (\lambda b.Z)[a{\mapsto}X] & = & \lambda b.(Z[a{\mapsto}X]) \\
(\mathbf{ren}{\mapsto}) & b\#Z \vdash & Z[a{\mapsto}b] & = & (b\ a)\cdot Z
\end{array}
$$

Theory of first-order logic FOL

Assume term-formers $=, \forall, \Rightarrow, \bot,$ sub, and sugar. Usual rules for boolean algebra, plus axioms for substitution, plus:

$$
\begin{array}{rll}
(\forall\mathbf{L}) & \forall[a]P \Rightarrow P[a\mapsto T] & = \top \\
(\forall\wedge) & \forall[a](P \wedge Q) \Leftrightarrow \forall[a]P \wedge \forall[a]Q & = \top \\
(\forall\mathbf{R}) & a\#P \ \vdash\ \forall[a](P \Rightarrow Q) \Leftrightarrow (P \Rightarrow \forall[a]Q) & = \top \\
 & & \\
(=\mathbf{L}) & U = T \wedge P[a\mapsto T] \Rightarrow P[a\mapsto U] & = \top \\
(=\mathbf{R}) & T = T & = \top
\end{array}
$$

Freshness derivation rules

Read $a\#t$ as 'a is fresh for t'.

$$\frac{}{a\#b}\,(\#\mathbf{ab}) \qquad \frac{a\#t_1 \quad \cdots \quad a\#t_n}{a\#\mathsf{f}(t_1,\ldots,t_n)}\,(\#\mathbf{f})$$

$$\frac{}{a\#[a]t}\,(\#[]\mathbf{a}) \qquad \frac{a\#t}{a\#[b]t}\,(\#[]\mathbf{b}) \qquad \frac{\pi^{-1}(a)\#X}{a\#\pi X}\,(\#\mathbf{X})$$

Note this is syntax-directed. $a\#t$ is a side-condition, not a logical assertion (its validity is not influenced by axioms or valuations).

For example

$$\cfrac{\cfrac{\cfrac{}{a\#b}\,(\#\mathbf{ab})}{a\#[b]b}\,(\#[]\mathbf{b})}{a\#\lambda[b]b}\,(\#\mathsf{f})$$

$$\cfrac{a\#X \qquad \cfrac{\cfrac{}{a\#[a]Y}\,(\#[]\mathbf{a})}{a\#\lambda[a]Y}\,(\#\mathsf{f})}{a\#X(\lambda[a]Y)}\,(\#\mathsf{f})$$

For example

The following freshnesses cannot be derived:

$$\vdash a\#a \qquad \vdash a\#X(\lambda[a]Y) \qquad \vdash a\#(\lambda[a]b)a$$

Equality derivation rules

$$\frac{}{t=t}\,(\mathbf{refl}) \qquad \frac{t=u}{u=t}\,(\mathbf{symm}) \qquad \frac{t=u \quad u=v}{t=v}\,(\mathbf{tran})$$

$$\frac{t=u}{[a]t=[a]u}\,(\mathbf{cong[]}) \qquad \frac{t=u}{\mathsf{f}(\dots,t,\dots)=\mathsf{f}(\dots,u,\dots)}\,(\mathbf{congf}) \qquad \frac{\begin{array}{c}[a\#X]\\ \vdots\\ t=u\end{array}}{t=u}\,(\mathbf{fr}) \quad (a\notin t,u)$$

$$\frac{\nabla^{\pi}\sigma}{t^{\pi}\sigma=u^{\pi}\sigma}\,(\mathbf{ax}_{\nabla\vdash\mathbf{t}=\mathbf{u}}) \qquad \frac{a\#t \quad b\#t}{(a\;b)\cdot t=t}\,(\mathbf{perm})$$

Example derivations in LAM

$$\frac{}{(\lambda[b]a)b = a[b\mapsto b]}\,(\mathbf{ax}_\beta) \qquad \frac{}{(\lambda[b]b)a = b[b\mapsto a]}\,(\mathbf{ax}_\beta)$$

The right derivation shows that substitution does not avoid capture, reflecting informal practice.

$$\frac{\dfrac{}{a\#b}\,(\#\mathbf{ab})}{\lambda[a](ba) = b}\,(\mathbf{ax}_\eta) \qquad \frac{a\#a}{\lambda[a](aa) = a}\,(\mathbf{ax}_\eta)$$

The left derivation is valid but the right one is not, because $a\#a$ is not derivable.

Example derivations in LAM

Atoms are interpreted equivariantly; they cannot be identified.

$$\frac{c\#X}{([c]Y)[c\mapsto X] = [c](Y[c\mapsto X])}\,(\mathbf{ax_{abs\mapsto}})$$

is not a valid instance of $(\mathbf{abs}\mapsto)$ on page 10 (even assuming $c\#X$) since permutations are bijective.

There is no π such that both $\pi(a) = c$ and $\pi(b) = c$.

Example derivations in LAM

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(\lambda[a]b)a = b[a\mapsto a]}\,(\mathbf{ax}_\beta) \qquad \dfrac{\dfrac{}{a\#b}\,(\#\mathbf{ab})}{b[a\mapsto a] = b}\,(\mathbf{ax}_{\#\mapsto})}{(\lambda[a]b)a = b}\,(\mathbf{tran})}{((\lambda[a]b)a)a = ba}\,(\mathbf{congf})}{[a](((\lambda[a]b)a)a) = [a](ba)}\,(\mathbf{cong[]})}{\lambda[a](((\lambda[a]b)a)a) = \lambda[a](ba)}\,(\mathbf{congf}) \qquad \dfrac{\dfrac{}{a\#b}\,(\#\mathbf{ab})}{\lambda[a](ba) = b}\,(\mathbf{ax}_\eta)}{\lambda[a](((\lambda[a]b)a)a) = b}\,(\mathbf{tran})$$

The extra power of the rule $(\mathbf{fr})$

Consider a theory C with one axiom $a\#X \vdash X = a$. We can derive $X = Y$ with $(\mathbf{fr})$, but not without it.

$$
\dfrac{
  \dfrac{
    \dfrac{[a\#X]^1}{X = a}\,(\mathbf{ax_{a\#X\vdash X=a}})
    \qquad
    \dfrac{
      \dfrac{[a\#Y]^1}{Y = a}\,(\mathbf{ax_{a\#X\vdash X=a}})
    }{a = Y}\,(\mathbf{symm})
  }{X = Y}\,(\mathbf{tran})
}{X = Y}\,(\mathbf{fr})^1
$$

Derivation of $X[a{\mapsto}a] = X$ in SUB

There’s no axiom for this in SUB, but it can be done by renaming:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a\#[a]X}\,(\#[]\mathbf{a}) \qquad \dfrac{[b\#X]^1}{b\#[a]X}\,(\#[]\mathbf{b})}{[b](b\,a)\cdot X = [a]X}\,(\mathbf{perm})}{[a]X = [b](b\,a)\cdot X}\,(\mathbf{symm})}{X[a{\mapsto}a] = ((b\,a)\cdot X)[b{\mapsto}a]}\,(\mathbf{congf}) \qquad \dfrac{\dfrac{[b\#X]^1}{a\#(b\,a)\cdot X}\,(\#\mathbf{X})}{((b\,a)\cdot X)[b{\mapsto}a] = X}\,(\mathbf{ax}_{\mathbf{ren}\mapsto})}{X[a{\mapsto}a] = X}\,(\mathbf{tran})}{X[a{\mapsto}a] = X}\,(\mathbf{fr})^1$$

Nominal algebra

Why is this important? Nominal algebra is:

- An algebraic theory with names.
- A theory of truth at the informal meta-level.

It turns out there is an overlap between these two things; a (algebraic) theory of names is a good theory of the informal meta-level.

Over the past century, first/higher-order logic, λ-calculus, π-calculus, have been invented/developed. A mathematical language has evolved to describe them. This mathematical language has two levels of variable, and freshness conditions. Nominal algebra is the formal algebraic version of that language.

This seems to me a unique mathematical opportunity.

Meta-theory

Soundness and completeness of equality with respect to a generic semantics in nominal sets. (A model of a theory is a nominal set equipped with operators to interpret the term-formers, validating the axioms.)

Completeness of specific theories with respect to specific open term models (λ-calculus, substitution). This expresses a precise sense in which the theories SUB, LAM, FOL are ‘the truth, the whole truth, and nothing but the truth’.

A nominal HSP theorem.

Nominal HSP theorem

A basic result of universal algebra states a bijection between algebraic theories, and classes of models closed under Homomorphism, Subalgebra, and Product.

It’s a ‘prime number factorisation’ result, for algebra. It’s useful for proving positive and negative results. For example the theory ‘my model has two elements’ can’t be expressed in universal algebra, since the class of two-element sets is not closed under products.

Nominal HSP is the same, but for Homomorphism, Subalgebra, Product and Atoms-Abstraction (a specific nominal construction).

Further work

Refine nominal algebra to a (nominal) logical framework, within which traditional logic and calculi are easily axiomatised and reasoned about. With respect to simple type theory and higher-order logic, this is likely to have the advantage of better computational behaviour and a more elementary proof-theory and model-theory.

Work out reprsentation theorems for first-order logic, λ-calculus, and π-calculus. That is, what are the generators under HSPA of the class of models of particular well-known theories — cf. Antonino Salibra's 'lattice of lambda-theories'.

λ-abstract over X to obtain a two-level higher-order logic.