Nominal algebra with applications

Murdoch J Gabbay

Tel-Aviv University, Israel 4/5/2006

Thanks to Nachum Dershowitz and Arnon Avron

This talk. . .

. . . is the first in a series of about four talks I plan to give in the framework of a mini-course describing (some? most?) of the mathematics I've done over the past six years (since I got my PhD).

Thank you to all of you for coming, and a special thank-you to those who arrived from out of town. I appreciate the interest!

Motivation

In this talk I'll motivate, present, and discuss Nominal Algebra. Thanks to Anna for the conversation on which the structure of this talk is based.

Motivation

Let’s look at some things we often write:

Motivation

- $\lambda a.t$ — untyped λ-calculus (LAM)
- $\forall a.\phi$ — first-order predicate logic (FOL)
- $\int \mathrm{f}\, da$ — school/kindergarten
- $\phi[a \mapsto t]$ — FOL (detail of $\forall$-rule)
- $u[a \mapsto t]$ — LAM (detail of β-rule)

These expressions all have in common:

- An object-level variable a.
- Meta-level variable symbols such as t, u, ϕ, or f.

(The detail)

$$\frac{\Gamma, \phi[a \mapsto t] \vdash \psi}{\Gamma, \forall a.\phi \vdash \psi} \quad (\forall L)$$

$$(\lambda a.u)t \to u[a \mapsto t] \qquad (\beta).$$

Motivation

Let’s build a logic which explicitly represents this (and in which LAM, FOL, and similar systems, are object-theories).

Let’s make the logic an algebra (for simplicity).

Let’s call it Nominal Algebra.

Nominal Terms

Nominal terms are a syntax inductively generated by

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

Here:

- We fix $a, b, c, \ldots \in \mathbb{A}$ a countably infinite set of atoms.
- We fix $X, Y, Z, \ldots \in \mathbb{V}$ a countably infinite set of unknowns (disjoint from the atoms; everything's disjoint).
- We fix f, g, … some term-formers.
- Call $[a]t$ an abstraction.

Nominal Terms

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

π is a permutation. A permutation is a finitely supported bijection on $\mathbb{A}$.

Finitely supported means:

$\pi(a) = a$ for all $a \in \mathbb{A}$ except for a finite set of atoms.

Nominal Terms

For example permutations are:

$$(a\ b\ c) \quad \text{and} \quad \textbf{Id}$$

(a to b to c to a, and the identity function). Permutations are not:

$$(a_1\ a_2)(a_3\ a_4)\ldots$$

for $\mathbb{A} = \{a_1, a_2, \ldots\}$.

Questions

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

Q. Why the a? Are they like variable symbols?

A. They represent object-level variable symbols.

- $\lambda a.t$ — untyped λ-calculus (LAM)
- $\forall a.\phi$ — fi rst-order predicate logic (FOL)
- $\int \mathrm{f}\, d\, a$ — school/kindergarten
- $\phi[a \mapsto t]$ — FOL (detail of $\forall$-rule)
- $u[a \mapsto t]$ — LAM (detail of β-rule)

Questions

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

Q. What is abstraction $[a]t$?

A. This represents abstract a in t.

- $\lambda[a]t$ untyped λ-calculus (LAM)
- $\forall[a]\phi$ fi rst-order predicate logic (FOL)
- $\int [a]\mathrm{f}\, d$ school/kindergarten
- $([a]\phi)[\mapsto t]$ FOL (detail of $\forall$-rule)
- $([a]u)[\mapsto t]$ LAM (detail of β-rule)

Questions

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

λ, $\forall$, $\int d$, and $[\mapsto]$ are represented by unary, unary, unary, and binary term-formers.

- $\lambda[a]t$ — untyped λ-calculus (LAM)
- $\forall[a]\phi$ — fi rst-order predicate logic (FOL)
- $\int[a]\mathrm{f}\,d$ — school/kindergarten
- $\Sigma([a]\phi,\ t)$ — FOL (detail of $\forall$-rule)
- $\Sigma([a]u,\ t)$ — LAM (detail of β-rule)

Questions

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

t, u, and ϕ, in the box of the previous slide are represented by unknowns in the syntax above; use capital letters for unknowns. Write $u[a \mapsto t]$ for $\Sigma([a]u,\ t)$.

- $\lambda[a]X$ — untyped λ-calculus (LAM)
- $\forall[a]P$ — fi rst-order predicate logic (FOL)
- $\int[a]X\,d$ — school/kindergarten
- $P[a \mapsto T]$ — FOL (detail of $\forall$-rule)
- $U[a \mapsto T]$ — LAM (detail of β-rule)

Internalising

We have internalised part of the meta-level. We can still use a meta-level if we like, i.e. we can still let t vary over unknown terms, but now terms are enriched with X which in the syntax represents an unknown term, as well as with $[a]X$ which in the syntax represents 'abstract a in X'.

- Object-level variables are modelled by atoms a.
- Meta-level unknowns are modelled by unknowns X.

Equality assertions $t = u$

We're doing algebra, so introduce a judgement for $t = u$, which judges whether the terms t and u are equal.

We called $[\text{-}]$- abstraction.

So intuitively we expect $[a]a = [b]b$ to hold — they are not identical syntax, but they are equal. Similarly for $[a]c = [b]c$. But $[a]a \neq [b]a$. And so on.

Complications

But we lose naïve α-equivalence. For example

$$[a]X = [b]X$$

ceases to hold, even though a and b are not in X, because the meaning of X is an 'unknown', and we do not know whether X mentions a or b! If we instantiate X to a, b, and c, we get respectively

$$[a]a \neq [b]a \qquad [a]b \neq [b]b \qquad [a]c = [b]c.$$

Note also we must introduce $\Sigma([a]Y, X)$ (sugared to $Y[a{\mapsto}X]$) because Y and X represent unknown terms, and 'until we know what they are' we have no way to carry out the substitution.

Freshness assertions $a\#t$

Read $a\#X$ as 'a does not occur in X', or 'a is fresh for X'.

Then we can characterise α-equivalence as:

$$b\#X \Rightarrow [b](b\ a)X = [a]X.$$

For the moment I'm just telling you that this is the case.

Call a pair $a\#t$ a freshness assertion. If $t \equiv X$ call it primitive.

Freshness derivation rules (formally)

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

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

Core equality derivation rules (formally)

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

$$\frac{t = u}{C[t] = C[u]}\,(cong) \qquad \frac{a\#t \quad b\#t}{(a\ b)\cdot t = t}\,(perm)$$

For example

$$\dfrac{\dfrac{}{a\#b}(\#ab) \qquad \dfrac{}{a\#b}(\#ab)}{[a]a = [b]b}(perm)$$

$$(b\ a) \cdot [b]b \equiv [a]a$$

$$\dfrac{\dfrac{b\#X}{b\#[a]X}(\#[]a) \qquad \dfrac{}{a\#[a]X}(\#[]a)}{[b](b\ a)X = [a]X}(perm)$$

$$(b\ a) \cdot [a]X \equiv [b](b\ a)X$$

Here $\equiv$ is syntactic identity.

Axioms

A freshness context Δ is a finite set of primitive freshness assertions.

An axiom $\Delta \vdash t = u$ is a pair of a freshness context and an equality assertion. If Δ is empty write it just $t = u$.

We can use axioms to enrich provable equality, which currently stands at some generalisation of α-equivalence.

Theory of λ-calculus LAM

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

(Assume suitable term-formers λ, app and sugar.)

As an axiom, we instantiate Y and X to 'any term' when we enrich equality, generating a family of equalities for each instantiation (and each context). Thus, Y and X do represent 'any term', with universal quantification at top level. Instantiation is direct replacement of an unknown by a term (no capture avoidance).

Theory of first-order logic FOL

$$P \Rightarrow Q \Rightarrow P \;=\; \top \qquad (P \Rightarrow Q) \Rightarrow (Q \Rightarrow R) \Rightarrow (P \Rightarrow R) \;=\; \top$$

$$\neg\neg P \Rightarrow P \;=\; \top \qquad \forall[a](P \wedge Q) \Leftrightarrow \forall[a]P \wedge \forall[a]Q \;=\; \top$$

$$\bot \Rightarrow P \;=\; \top \qquad a\#P \;\vdash\; \forall[a](P \Rightarrow Q) \Leftrightarrow (P \Rightarrow \forall[a]Q) \;=\; \top$$

$$T \approx T \;=\; \top \qquad \forall[a]P \Rightarrow P[a\mapsto T] \;=\; \top$$

$$U \approx T \wedge P[a\mapsto T] \Rightarrow P[a\mapsto U] \;=\; \top$$

(Assume suitable term-formers $\approx, \forall, \Rightarrow, \bot$ and sugar.)

The '$= \top$' bit just converts a predicate into a judgement.

But wait. . .

Remember that substitution is a term-former.

Theory of substitution SUB

$$\mathsf{f}(X_1, \ldots, X_n)[a \mapsto T] = \mathsf{f}(X_1[a \mapsto T], \ldots, X_n[a \mapsto T])$$

$$b\#T \vdash ([b]X)[a \mapsto T] = [b](X[a \mapsto T])$$

$$a[a \mapsto T] = T$$

$$a\#X \vdash X[a \mapsto T] = X$$

$$b\#X \vdash X[a \mapsto b] = (b\ a)X$$

Picture of what we have done

- $\equiv$ is syntactic identity $\quad [a]a \not\equiv [b]b$
- $=$ (no axioms) is α-equivalence $\quad b\#X \vdash [b](b\ a)X = [a]x$
- $=_{\mathsf{SUB}}$ is substitution $\quad b\#Y \vdash Y[b\mapsto X] = Y$
- $=_{\mathsf{LAM}}$ is $\alpha\beta$-equivalence $\quad (\lambda[a]a)b = b$
- $=_{\mathsf{FOL}}$ is logical equivalence $\quad (\forall[a](a \approx a)) = \top$

There is much to say about all of these theories. In another talk I will discuss SUB.

We concentrate on FOL. . .

. . . in the rest of this talk. We build/specify FOL on top of SUB, so that as far as it is concerned, substitution (and α-equivalence) are structural properties of formulae.

That formulae may contain explicit ‘unknown formulae’ is no more (or less!) relevant to the system, than it is to us when we write $\forall a.\phi$.

Nominal algebra theories

This system can make formal assertions about unknown formulae and relations between them, in the syntax itself, because we have the syntax to do it; X, Y, Z.

This means we get extra: not only first-order logic but one-and-a-halfth-order logic, which is first-order logic whose syntax is enriched with first-class formula unknowns.

The treatment of binding is significantly different from that of second-order logic.

Remember: once we have built something in a framework (Nominal Algebra), we can throw away the framework. That's what we do next.

Recall FOL

$$
\begin{array}{rclcrcl}
P \Rightarrow Q \Rightarrow P & = & \top & \quad & (P \Rightarrow Q) \Rightarrow (Q \Rightarrow R) \Rightarrow (P \Rightarrow R) & = & \top \\
\neg\neg P \Rightarrow P & = & \top & & \forall[a](P \wedge Q) \Leftrightarrow \forall[a]P \wedge \forall[a]Q & = & \top \\
\bot \Rightarrow P & = & \top & & a\#P \vdash \forall[a](P \Rightarrow Q) \Leftrightarrow P \Rightarrow \forall[a]Q & = & \top \\
T \approx T & = & \top & & \forall[a]P \Rightarrow P[a \mapsto T] & = & \top \\
 & & & & U \approx T \wedge P[a \mapsto T] \Rightarrow P[a \mapsto U] & = & \top
\end{array}
$$

Soon we'll have a sequent system corresponding to this particular Nominal Algebra theory.

Recall: First-Order Logic (FOL)

Fix countably infinitely many variable symbols $a, b, c, \ldots$. Let terms be:

$$t ::= a$$

Formulae or predicates are:

$$\phi ::= \bot \mid \phi \Rightarrow \phi \mid \forall a.\phi \mid t \approx t'.$$

Write $\equiv$ for syntactic identity.

Derivation

A context Φ and cocontext Ψ are finite and possibly empty sets of formulae. A judgement is a pair $\Phi \vdash \Psi$. Valid judgements:

$$(Axiom)\ \frac{}{\phi,\ \Phi \vdash \Psi,\ \phi} \qquad (\bot L)\ \frac{}{\bot,\ \Phi \vdash \Psi}$$

$$(\Rightarrow R)\ \frac{\phi,\ \Phi \vdash \Psi,\ \psi}{\Phi \vdash \Psi,\ \phi \Rightarrow \psi} \qquad (\Rightarrow L)\ \frac{\Phi \vdash \Psi,\ \phi \quad \psi,\ \Phi \vdash \Psi}{\phi \Rightarrow \psi,\ \Phi \vdash \Psi}$$

$$(\forall R)\ \frac{\Phi \vdash \Psi,\ \psi}{\Phi \vdash \Psi,\ \forall a.\psi}\ a \text{ fresh for } \Phi,\Psi \qquad (\forall L)\ \frac{\phi[a\mapsto t],\ \Phi \vdash \Psi}{\forall a.\phi,\ \Phi \vdash \Psi}$$

Hang on a moment

What are ϕ and ψ?

Meta-variables ranging over formulae.

What are t and a?

Meta-variables ranging over terms and variable symbols.

What is $\phi[a\mapsto t]$?

A meta-level operation defined given real predicate, variable symbol, and term.

What is 'a fresh for Φ and Ψ'?

A meta-level condition defined given a real context and cocontext.

Schema

Quite a lot of things happen in the meta-level in First-Order Logic (FOL). For example

$$\vdash \forall a.\forall b.\phi \Leftrightarrow \forall b.\forall a.\phi$$

is derivable for every value of the meta-variable ϕ:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\phi \vdash \phi}(Axiom)}{\forall b.\phi \vdash \phi}(\forall L)}{\forall a.\forall b.\phi \vdash \phi}(\forall L)}{\forall a.\forall b.\phi \vdash \forall a.\phi}(\forall R)}{\forall a.\forall b.\phi \vdash \forall b.\forall a.\phi}(\forall R)$$

Schema

However, the fact that this happens for all ϕ cannot be expressed in FOL.

Some nice example theorems:

- If $t \approx t'$ then $\phi[a \mapsto t] \Leftrightarrow \phi[a \mapsto t']$.
- If $a \not\in fv(\phi)$ then $\vdash (\forall a.\phi) \Leftrightarrow \phi$.
- $\forall a.\forall b.\phi$ if and only if $\forall b.\forall a.\phi$.

Second/Higher-order logic

In Higher-Order Logic (HOL), propositions have a type o and $\forall_\sigma$ is a constant with type $(\sigma \longrightarrow o) \longrightarrow o$, write just $\forall$ or $\forall : (\sigma \longrightarrow o) \longrightarrow o$.

Then the single sequent

$$\vdash \forall\lambda f.\big(\forall\lambda a.\forall\lambda b.fab \Leftrightarrow \forall\lambda b.\forall\lambda a.fab\big)$$

expresses that

$$\vdash \forall a.\forall b.\phi \Leftrightarrow \forall b.\forall a.\phi$$

holds for all ϕ.

Here f has function type. If $a : \sigma$ and $b : \tau$ then $f : \sigma \longrightarrow \tau \longrightarrow o$ and 'fab is ϕ'.

Second/Higher-order logic

Similarly:

- 'If $t \approx t'$ then $\phi[a \mapsto t] \Leftrightarrow \phi[a \mapsto t']$' becomes

$$t \approx t' \vdash \forall \lambda f. \left(ft \Leftrightarrow ft'\right).$$

  in HOL.

  Note the types: f has function type and if $t : \sigma$ then $f : \sigma \longrightarrow o$ and $\forall : ((\sigma \longrightarrow o) \longrightarrow o) \longrightarrow o$.

- 'If $a \notin fv(\phi)$ then $\vdash \forall a.\phi \Leftrightarrow \phi$' is not expressible in HOL.

Schema

One-and-a-halfth order logic addresses these problems in a different way.

Take term-formers $\approx$, $\forall$, $\Rightarrow$, and $\bot$.

Sugar and example terms

Write $\neg\phi$ for $\phi \Rightarrow \bot$, write $\phi \wedge \phi'$ for $\neg(\phi \Rightarrow \neg\phi')$, write $\phi \Leftrightarrow \phi'$ for $(\phi \Rightarrow \phi') \wedge (\phi' \Rightarrow \phi)$, write $\phi \vee \phi'$ for $(\neg\phi) \Rightarrow \phi'$, write $\top$ for $\bot \Rightarrow \bot$.

- $\forall[a]\forall[b]X \Leftrightarrow \forall[b]\forall[a]X$.
- $T \approx T'$.
- $X[a \mapsto T] \Leftrightarrow X[a \mapsto T']$.
- $\forall[a]X \Leftrightarrow X$.

Sequent derivation rules

$$\frac{}{\phi,\ \Phi \vdash_{\triangle} \Psi,\ \phi}\,(Axiom) \qquad \frac{}{\bot,\ \Phi \vdash_{\triangle} \Psi}\,(\bot L)$$

$$\frac{\Phi \vdash_{\triangle} \Psi,\ \phi \qquad \psi,\ \Phi \vdash_{\triangle} \Psi}{\phi \Rightarrow \psi,\ \Phi \vdash_{\triangle} \Psi}\,(\Rightarrow L) \qquad \frac{\phi,\ \Phi \vdash_{\triangle} \Psi,\ \psi}{\Phi \vdash_{\triangle} \Psi,\ \phi \Rightarrow \psi}\,(\Rightarrow R)$$

$$\frac{\phi',\ \Phi \vdash_{\triangle} \Psi \qquad \Delta \vdash_{\mathsf{SUB}} \phi' = \phi[a \mapsto t]}{\forall[a]\phi,\ \Phi \vdash_{\triangle} \Psi}\,(\forall L)$$

$$\frac{\Phi \vdash_{\triangle} \Psi,\ \psi \qquad \Delta \vdash a\#\Phi, \Psi}{\Phi \vdash_{\triangle} \Psi,\ \forall[a]\psi}\,(\forall R)$$

Em. . . just a few more sequent derivation rules

$$\frac{}{\Phi \vdash \Psi,\ t \approx t}\ (\approx R)$$

$$\frac{\phi',\ \Phi \vdash \Psi \qquad \Delta \vdash_{\mathsf{SUB}} \phi' = \phi''[a{\mapsto}t'] \qquad \Delta \vdash_{\mathsf{SUB}} \phi = \phi''[a{\mapsto}t]}{t'{\approx}t,\ \phi,\ \Phi \vdash_{\Delta} \Psi}\ (\approx L)$$

$$\frac{\phi',\ \Phi \vdash_{\Delta} \Psi \qquad \Delta \vdash_{\mathsf{SUB}} \phi' = \phi}{\phi,\ \Phi \vdash_{\Delta} \Psi}\ (StructL)$$

$$\frac{\Phi \vdash_{\Delta} \Psi,\ \psi' \qquad \Delta \vdash_{\mathsf{SUB}} \psi' = \psi}{\Phi \vdash_{\Delta} \Psi,\ \psi}\ (StructR)$$

Example derivations

$$\dfrac{\dfrac{\forall[a]\forall[b]X \vdash X \qquad a\#\forall[b]X}{\forall[a]\forall[b]X \vdash \forall[a]X}(\forall R) \qquad b\#\forall[a]\forall[b]X}{\forall[a]\forall[b]X \vdash \forall[b]\forall[a]X}(\forall R)$$

$$\dfrac{\dfrac{\dfrac{}{X \vdash X}(Axiom) \qquad \vdash_{\mathsf{SUB}}\ X = X[b\mapsto b]}{\forall[b]X \vdash X}(\forall L) \qquad \vdash_{\mathsf{SUB}}\ \forall[b]X = (\forall[b]X)[a\mapsto a]}{\forall[a]\forall[b]X \vdash X}(\forall L)$$

Semantics in FOL: "For all ϕ and ψ, $\forall a.\forall b.\phi \vdash \forall b.\forall a.\psi$."

Freshness part of the derivation

$$
\cfrac{\cfrac{\cfrac{\cfrac{}{b\#[b]X}\,(\#[]a)}{b\#\forall[b]X}\,(\#\mathsf{f})}{b\#[a]\forall[b]X}\,(\#[]a)}{b\#\forall[a]\forall[b]X}\,(\#\mathsf{f})
$$

Another example derivation

$$\dfrac{\dfrac{}{X[a\mapsto T'] \vdash X[a\mapsto T']}(Axiom) \quad \begin{array}{c}\vdash_{\mathsf{SUB}} X[a\mapsto a][a\mapsto T'] = X[a\mapsto T'], \\ \vdash_{\mathsf{SUB}} X[a\mapsto a][a\mapsto T] = X[a\mapsto T]\end{array}}{T' \approx T,\ X[a\mapsto T] \vdash X[a\mapsto T']}(\approx L)$$

Semantics in FOL:

"For all t and t' and ϕ, $\quad t' \approx t,\ \phi[a\mapsto t] \vdash \phi[a\mapsto t']$."

One more example derivation

$$\dfrac{\dfrac{}{X \vdash_{a\#X} X}(Axiom) \qquad a\#X \vdash a\#X}{X \vdash_{a\#X} \forall[a]X}(\forall R)$$

Semantics in FOL:

"For all ϕ and a, if $a \notin fv(\phi)$ then $\phi \vdash \forall a.\phi$."

A nice theorem:

$$\frac{\Phi \vdash_{\triangle} \Psi, \phi \qquad \phi, \Phi \vdash_{\triangle} \Psi}{\Phi \vdash_{\triangle} \Psi}(Cut)$$

Theorem (cut-elimination): Cut is eliminable.

The cut-elimination procedure is almost standard — but this is cut-elimination in the presence of unknown formulae.

Since the cut-elimination procedure is normally written parametrically over those formulae, this is no surprise really. However, the meta-level reasoning about substitution and α-equivalence is now all completely explicit on the nominal terms.

Another nice theorem:

Say a nominal term is closed when it mentions no unknowns. So a is closed but X is not.

Theorem: First-order logic (and its derivations) correspond to sequents of closed terms (and their derivations); term-for-term up to $\vdash_{\mathsf{SUB}}$, and proof-rule by proof-rule (up to $(Struct)$).

Publicity for $\vdash^{\mathsf{SUB}}$ (next talk, probably)

Write $t \vdash^{\mathsf{SUB}}_{\Delta} u$ when $t = u$ is derivable from assumptions Δ using the following axioms:

$$
\begin{array}{rrcl}
(f\mapsto) & \mathsf{f}(u_1,\ldots,u_n)[a\mapsto t] & = & \mathsf{f}(u_1[a\mapsto t],\ldots,u_n[a\mapsto t]) \\
([b]\mapsto) & b\#t \Rightarrow ([b]u)[a\mapsto t] & = & [b](u[a\mapsto t]) \\
(var\mapsto) & a[a\mapsto t] & = & t \\
(u\mapsto) & a\#u \Rightarrow u[a\mapsto t] & = & u \\
(ren\mapsto) & b\#u \Rightarrow u[a\mapsto b] & = & (b\ a)\cdot u \\
(perm) & a,b\#t \Rightarrow (a\ b)\cdot t & = & t
\end{array}
$$

SUB

What is the theory of substitution? Is it decidable? What are its models? How do we know we have the right axioms?

Permutation action

$$\pi \cdot a \equiv \pi(a) \qquad \pi \cdot (\pi' X) \equiv (\pi \circ \pi')X$$

$$\pi \cdot [a]t \equiv [\pi(a)](\pi t)$$

$$\pi \cdot \mathsf{f}(t_1, \ldots, t_n) \equiv \mathsf{f}(\pi t_1, \ldots, \pi t_n)$$

Relation to HOL

Not direct since we can express $a\#t$ and HOL cannot.

Also, suppose $X : o$ and $t : \mathbb{T}$. Then $X[a{\mapsto}t]$ corresponds to ft in HOL where $f : \mathbb{T} \to o$. However, $X[a{\mapsto}t][a'{\mapsto}t']$ corresponds to $f'tt'$ where $f' : \mathbb{T} \to \mathbb{T} \to o$. Similarly $X[a{\mapsto}t][a'{\mapsto}t'][a''{\mapsto}t'']\ldots$

This is type raising.

In one-and-a-halfth-order logic, X remains at sort o throughout and the universal quantification implicit in the use of X allows arbitrary numbers of substitutions.

Relation to HOL

One-and-a-halfth-order logic is not fully higher-order. We can write

$$X \vdash Y$$

meaning in FOL "For all formulae ϕ and ψ, $\phi \vdash \psi$."

In HOL we can write this as $\vdash \forall \phi, \psi . \phi \Rightarrow \psi$.

However we can also write $\vdash \forall \psi . \big((\forall \phi . \phi) \Rightarrow \psi \big)$.

This is not possible in one-and-a-halfth-order logic: $(\forall [X] X) \vdash Y$ is not syntax.

Conclusions

Nominal Algebra enriches algebra with object-level variable symbols (atoms) with primitive facilities for abstraction and α-renaming ($[a]t$, πX, $a\#X$).

We can use this theory of axiomatise systems with binding, like first-order logic.

We thus get a formal framework for defining logics (and calculi).

Conclusions

Once your theory is specified, you can throw out the framework and just keep the theory . . .

. . . with nominal algebra providing now a semantics, e.g. the sequent system for one-and-a-halfth-order logic has a sound and complete semantics in FOL, with

$$\Phi \vdash_{\Delta} \Psi \quad \text{translating to} \quad \Delta \vdash (\Phi^{\wedge} \Rightarrow \Psi^{\vee}) = \top.$$

($^{\wedge}$ means 'put $\wedge$ between the elements of Φ', similarly for $^{\vee}$).

Conclusions

We get extra. E.g. one-and-a-halfth order logic was intended to be first-order logic, then we noticed that we had predicate unknowns; thus enabling us to reason universally on predicates in a new way.

This really is new, because $a\#X$ is not expressible using other techniques (to our knowledge); not in full generality for a completely unkown X.

Conclusions

For some further work, how about...

- Two-and-a-halfth-order logic (where you can abstract X)?
- Implementation and automation?
- Semantics (aside from in FOL)?
- Axiomatisations of other logics. Remember: we can always build an 'onion' and delegate structural matters such as $=_{\mathsf{SUB}}$ to structural rules. We can do this even in the presence of unknowns. That is the point:

Structure and abstraction in the presence of first-class unknowns.