# Nominal: the big picture

Murdoch J. Gabbay

29 April 2016

# Introduction

Thanks to Neel Krishnaswami organising this talk.

Thanks to the Theory group for inviting me.

Thank you all for coming.

# An example: sigma

Nominal algebra (Gabbay & Mathijssen 2006) is like universal algebra but over nominal sets, so enriched with nominal-style names, freshness, and binding.

Let's look at an axiomatisation of substitution $Z[a{\mapsto}X]$:

$$\begin{array}{lrl} a\#Z \Rightarrow & Z[a{\mapsto}X] & = Z \\ & Z[a{\mapsto}a] & = Z \\ a\#Y \Rightarrow & Z[a{\mapsto}X][b{\mapsto}Y] & = Z[b{\mapsto}Y][a{\mapsto}X[b{\mapsto}Y]] \\ b\#Z \Rightarrow & Z[a{\mapsto}X] & = ((b\ a){\cdot}Z)[b{\mapsto}X] \end{array}$$

Call a set $\mathbb{T}$ with an operation $\sigma$ satisfying the axioms above, a sigma-algebra.

Substitution is an operation of type $\mathbb{T} \times \mathbb{A} \times \mathbb{T} \to \mathbb{T}$ on a nominal set $\mathbb{T}$, satisfying the axioms above.

Just like group multiplication has type $G \times G \to G$ or logical conjunction has type $B \times B \to B$.

# Sigma

$$
\begin{array}{lrl}
a\#Z \Rightarrow & Z[a \mapsto X] & = Z \\
 & Z[a \mapsto a] & = Z \\
a\#Y \Rightarrow & Z[a \mapsto X][b \mapsto Y] & = Z[b \mapsto Y][a \mapsto X[b \mapsto Y]] \\
b\#Z \Rightarrow & Z[a \mapsto X] & = ((b\ a) \cdot Z)[b \mapsto X]
\end{array}
$$

$a\#Z$ is a freshness side-condition. It corresponds to saying 'if $a$ is not free in $Z$'.

$(b\ a) \cdot Z$ is a permutation. It corresponds to 'swap $b$ and $a$ in $Z$'.

Both have natural interpretions over nominal sets.

As lemmas of the concrete syntactic model of syntactic substitution $:=$, the axioms above are often called: garbage-collection, identity, the substitution lemma, and $\alpha$-renaming.

# alpha $\lesssim$ sigma

Let's rewrite the final $\alpha$-renaming axiom as one might see it in a textbook (as a lemma, not an axiom):

$$\begin{array}{lll} b\#Z \Rightarrow & Z[a{\mapsto}X] = ((b\ a){\cdot}Z)[b{\mapsto}X] & \text{axiom} \\ b \notin fv(u) \Rightarrow & u[a{:=}s] = u[a{:=}b][b{:=}s] & \text{lemma} \\ b \notin fv(u) \Rightarrow & u[a{:=}s] = ((b\ a){\cdot}u)[b{:=}s] & \text{lemma} \end{array}$$

- The top line is an axiom in nominal algebra about $\sigma$ (i.e. $\mapsto$).
- The middle line is a lemma of concrete syntax and syntactic substitution $:=$.
- The bottom line is also a lemma, verifying validity of the top line if $\mathbb{T}$ is concrete syntax and $\mapsto$ is interpreted as $:=$.

If $b \notin fv(u)$ then $u[a{:=}b] = (b\ a){\cdot}u$, so we can say the middle line and the bottom line are, up to this little lemma, identical.

# alpha $\lesssim$ sigma

But there is a crucial difference! If we think in terms of rewriting,

- $u[a{:=}s] \to u[a{:=}b][b{:=}s]$ has two substitutions on the right, because we are managing $\alpha$-renaming using substitution which is the very thing we are trying to define.
- $u[a{:=}s] \to ((b\ a){\cdot}u)[b{:=}s]$ has only one substitution on the right, because we manage $\alpha$-renaming using permutations.

Permutations form a group and are invertible; substitutions form a monoid and are not.

A mathematical theory of $\alpha$-renaming can be based on the group of permutations. Substitution can be built on top of that.

There is no need to interleave the theory of $\alpha$-renaming with the theory of substitution.

I will say that alpha is decoupled from sigma. In symbols: $\alpha \lesssim \sigma$.

# Quantifiers

Assume a nominal lattice (that is, a lattice over nominal sets).
Then universal quantification can be axiomatised as follows:

$$\begin{aligned} b\#X \Rightarrow \quad & \forall a.X = \forall b.(b\ a)\cdot X \\ & \forall a.X \leq X \\ a\#X \Rightarrow \quad & X \leq \forall a.X \\ & \forall a.(X \wedge Y) = (\forall a.X) \wedge \forall a.Y \end{aligned}$$

Things to note:

- This is equivalent to the following universal property:

$$\forall a.X = \bigvee \{X' \leq X \mid a\#X'\}.$$

- Equivalent to 'adjoint' property:

$$a\#Y \Rightarrow Y \leq X \Leftrightarrow Y \leq \forall a.X$$

- We use freshness and renaming above, so we can write $\alpha \leq \forall$.
- We have decoupled from substitution. In symbols: $\sigma \not\leq \forall$.

# Closer look at quantifiers

Perhaps you're looking at the 'adjoint' characterisation:

$$a\#Y \Rightarrow Y \leq X \Leftrightarrow Y \leq \forall a.X$$

and comparing it with this, taken from p29 of "Introduction to Categorical Logic" by Awodey and Bauer, where $\varphi \leq B \times A$ and $\vartheta \leq B$:

$$\frac{\vartheta \times A \leq \varphi}{\vartheta \leq \forall_A \varphi} \qquad \text{resembles} \qquad \frac{Y \leq X \quad (a\#Y)}{Y \leq \forall a.X}$$

But there is a crucial difference! To interpret $\vartheta$ and $\varphi$ we must have objects $A$ and $B$ to hand, and a category including arrows for substitution and so forth.

# Closer look at quantifiers

The category theory presentation hides structure—as it is designed to—but the structure is still there.

In the nominal approach to quantification on which my recent papers are based, it is not the case that $\sigma$ is needed but perhaps elided. It is irrelevant—or even absent.

We may decide we want to use it (see next slide), but we don't need $\sigma$ to define $\forall$ or $\alpha$.

We can calculate $\forall a.X$ in a nominal lattice regardless of its $\sigma$-algebra structure, if it has any at all.

# Adding names

I use this decoupling twice:

- In the Stone dualities for first-order logic and the $\lambda$-calculus, the construction of points precedes the construction of the $\sigma$-action on predicates.
  In a certain critical sense, we build $\forall$ before we build $\sigma$. See Proposition 5.2.8 (eight characterisations of $\forall$) and Definition 6.1.1 of [gabbay:repdul].
- In my claimed ConNF proof, terms and predicates are interleaved (by comprehension: $\{a \mid \phi\}$ is a term if $\phi$ is a predicate).
  We can still give nominal semantics to $\forall$, even though we have 'not finished building' our universe of terms. See Figure 3 axiom (modall) of [gabbay:conqnf].

# The big picture (so far)

- Everything depends on $\alpha$, where $\alpha = \# + \pi$.
- $\alpha$ does not depend on $\sigma$:

$$\cancel{[a]X = [b](X[a \mapsto b])} \qquad [a]X = [b](b\ a)\cdot X.$$

- $\forall$ does not depend on $\sigma$:

$$\cancel{\forall a.X = \bigwedge_u X[a := U]} \qquad \forall a.X = \bigvee \{X' \leq X \mid a\#X'\}.$$

- $\sigma$ and $\forall$ do depend on $\alpha$:

$$X[a \mapsto U] = ((b\ a)\cdot X)[b \mapsto U] \qquad \forall a.X = \forall b.(b\ a)\cdot X.$$

# If there's time: representation of sigma-algebras

$$
\begin{array}{lrl}
a\#Z \Rightarrow & Z[a \mapsto X] & = Z \\
 & Z[a \mapsto a] & = Z \\
a\#Y \Rightarrow & Z[a \mapsto X][b \mapsto Y] & = Z[b \mapsto Y][a \mapsto X[b \mapsto Y]] \\
b\#Z \Rightarrow & Z[a \mapsto X] & = ((b\ a) \cdot Z)[b \mapsto X]
\end{array}
$$

# From sigma to amgis

Assume a sigma-algebra $x, y, z, u, v \in \mathcal{X}$.
Consider its powerset $p, q \in pow(\mathcal{X})$.

Define an amgis-action on sets by:

$$x \in p[u \leftarrow\!\shortmid a] \Leftrightarrow x[a \shortmid\!\rightarrow u] \in p \quad \text{so}$$
$$p[u \leftarrow\!\shortmid a] = \{x \mid x[a \shortmid\!\rightarrow u] \in p\}.$$

The amgis-action is the functional preimage of the sigma-action.

(Amgis-algebras can be axiomatised, as sigma-algebras were axiomatised above. See e.g. [gabbay:semooc].)

Think of $p[u \leftarrow\!\shortmid a]$ as

"$p$ reprogrammed to believe that $a$ is equal to $u$."

# From amgis back to sigma

Assume an amgis-algebra $p, q \in \mathcal{P}$.
Consider its powerset $X, Y \in pow(\mathcal{P})$.

Define an action by:

$$p \in X[a \mapsto u] \Leftrightarrow И b.p[u \leftarrow\!\shortmid b] \in (b\ a)\cdot X \quad \text{so}$$
$$X[a \mapsto u] = \{p \mid И b.(p[u \leftarrow\!\shortmid b] \in (b\ a)\cdot X)\}.$$

The $И$ is the new-quantifier (Gabbay & Pitts 1999). It means 'for a fresh name'.

This generates a sigma-algebra on $pow(\mathcal{P})$! Think of $X[a \mapsto u]$ as

"$X$ reprogrammed to believe that $a$ is equal to $u$ — then hide/bind $a$."

So taking powersets we alternate: sigma, amgis, sigma.

If $\mathcal{X}$ has sigma then $pow(\mathcal{X})$ has amgis, and $pow(pow(\mathcal{X}))$ has sigma and is also a nominal lattice.

# Equality

Pleasingly, $pow(pow(\mathcal{X}))$ can also interpret FOL equality:

$$u{=}v = \{p{\in}pow(\mathcal{X}) \mid И c.p[u{\leftarrowtail}c] = p[v{\leftarrowtail}c]\}.$$

$И c.p[u{\leftarrowtail}c] = p[v{\leftarrowtail}c]\}$ is a congruence property: it is precisely that which is necessary to derive

$$p \in X[a{\mapsto}u] \quad \text{if and only if} \quad p \in X[a{\mapsto}v].$$

Bearing in mind that $p \in X[a{\mapsto}u] \Leftrightarrow И c.(p[u{\leftarrowtail}c] \in (c\ a){\cdot}X)$.

# Another equality

If $\mathcal{X}$ has sigma then $pow(\mathcal{X})$ has amgis and $pow(pow(\mathcal{X}))$ has sigma, and is a nominal lattice.

So we can write the following beautiful equality:

$$\text{FOL} = \sigma + \text{powersets}.$$

This equality is what [gabbay:semooc] says.

Thus, if $\mathcal{X}$ is a $\sigma$-algebra then $pow(pow(\mathcal{X}))$ has rich sets structure:

- It has all the structure of a model of first-order logic.
- It is a sound and complete model of first-order logic.

We use this construction in the Stone duality result, based on the usual injection $X \in \mathcal{X} \longmapsto \{p \mid X \in p\} \in pow(pow(\mathcal{X}))$.

# Conclusions: the full big picture, so far

$\alpha = \pi + \#$

$\sigma \geq \alpha$

$\forall \geq \alpha$

$\forall$ and $\sigma$ are incomparable, but compatible.

$\mathfrak{o} = \sigma$

$= \geq \mathfrak{o} + И$

FOL $= \sigma +$ powersets.

$\lambda = \sigma +$ more stuff I haven't described.

NF $= \sigma + И + Э +$ lots more stuff I haven't even begun to describe.

More to follow, I am sure.

# Supplement: on duality

Consider Boolean algebra: the logic of $\wedge$, $\vee$, and $\neg$, satisfying axioms such as

$$\neg\neg\phi = \phi \quad \text{and} \quad \phi \wedge (\psi \vee \psi') = (\phi \vee \psi) \wedge (\phi \vee \psi').$$

Clearly conjunction $\wedge$ 'looks like' sets intersection $\cap$ and disjunction $\vee$ 'looks like' sets union $\cup$ and negation $\neg$ 'looks like' sets complement $\setminus$.

But is there some model of Boolean algebras that is so wild that it cannot be presented in these terms; so $\wedge$ cannot mean $\cap$ and $\vee$ cannot mean $\cup$ and $\neg$ cannot mean $\setminus$?

# Supplement: on duality

Stone duality for Boolean algebra says: no, there is no such model.

In fact, every Boolean algebra $\mathcal{B}$ can be presented as a subset of $pow(pow(\mathcal{B}))$ where $\wedge$ is $\cap$ and $\vee$ is $\cup$ and $\neg$ is $pow(pow(\mathcal{B})) \setminus$ -.

And every map of Boolean algebras extends to a map of this presentation.

(More technically: Boolean algebras correspond to compact totally disconnected Hausdorff spaces; maps of Boolean algebras correspond to continuous functions.)

Stone duality is a strong sanity guarantee, that logical symbols correspond to sets operations. It gives one very precise, fine-grained account of what a logic 'really means'.

A full Stone duality result is typically hard work ($\geq$50 pages). You don't need to understand the proof, to use the result!

# Supplement: on duality

In brief, my three most recent papers work by giving Stone representations/dualities for first-order logic, the $\lambda$-calculus, and set theory (modulo $\approx$90 pages of maths per paper!).

- No Stone duality result for the $\lambda$-calculus had previously been known. It was just not possible to engineer the proofs without the fine control of names given by nominal techniques.
- The set theory I consider, Quine's NF, could not be proved consistent using ordinary mathematics. It consistency has been an open problem since the 1930s.
  (NF is fascinating in its own right. This is for another talk.)

So this isn't just a new way of looking at old and well-understood systems.

The nominal models help us to see and prove new things we couldn't see and prove before.