Nominal Henkin Semantics: simply-typed lambda-calculus models in nominal sets

Murdoch J. Gabbay   Dominic P. Mulligan*

We investigate a class of nominal algebraic Henkin-style models for the simply typed λ-calculus in which variables map to names in the denotation and λ-abstraction maps to a (non-functional) name-abstraction operation. The resulting denotations are smaller and better-behaved, in ways we make precise, than functional valuation-based models.

Using these new models, we then develop a generalisation of λ-term syntax enriching them with existential meta-variables, thus yielding a theory of incomplete functions. This incompleteness is orthogonal to the usual notion of incompleteness given by function abstraction and application, and corresponds to holes and incomplete objects.

1 Introduction

In this paper we develop a Henkin-style semantics for the simply-typed λ-calculus in nominal sets. The simply-typed λ-calculus (STLC) has notions of typed variable, substitution, and function abstraction. Correspondingly, our models in nominal sets will enrich 'ordinary' sets with typed names, a substitution action, and name-abstraction. Thus, concepts that are normally characteristic of syntax—like variable, substitution, and variable-binding—are explicitly represented as nominal algebraic structure [22].

The resulting models have different properties from traditional valuation-style ('closed') semantics. Intuitively this is because leaving names in the denotation gives the models more structure—we have more information about 'where an element came from'.

For instance, Proposition 3.6 (the (ξ) rule) and Theorem 3.15 (well-pointedness) are properties that hold of the nominal models of this paper, and fail for 'classical' treatments (see Examples 3.7 and 3.16). This is because the direct inclusion of names into the denotation forces there to be 'enough' elements of the model, and naturality requirements of the models require these elements to be 'sufficiently distinguishable'. These conditions cannot be expressed without names in the denotation.

Furthermore, we find that we can extend this to a syntax and semantics for existential variables. That is, we will extend STLC syntax with 'holes'. The technique used is essentially the same as the *nominal terms* of [40] (a permissive variant thereof, following [6, 7]) but taking semantics in nominal models of STLC instead of in datatypes of abstract syntax with binding.

Because λ-abstraction maps to atoms-abstraction, the denotation of functions does not involve function spaces. Because variables map to themselves, valuations are not used either; their role is taken by the substitution for names. Thus we obtain a simple 'first-order flavoured' completeness proof (Theorem 3.11).

In summary, nominal Henkin models differ from 'ordinary' Henkin models by including variables and substitution in the underlying domain of the denotation as nominal algebraic

[*] We are very grateful to Peter Selinger and to two anonymous referees for their useful and constructive comments. We acknowledge the support of the Leverhulme trust. We acknowledge the support of grant RYC-2006-002131 at the Polytechnic University of Madrid. The project CerCo acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 243881.

H. Geuvers, G. Nadathur (Eds): LFMTP 2011
EPTCS 71, 2011, pp. 58–75, doi:10.4204/EPTCS.71.5

$$\frac{}{\Gamma, a{:}\phi \vdash a : \phi}\,(\mathbf{V}) \qquad \frac{(type(C) = \phi)}{\Gamma \vdash C : \phi}\,(\mathbf{C}) \qquad \frac{\Gamma, a{:}\phi \vdash r : \psi \ \ (a \notin dom(\Gamma))}{\Gamma \vdash (\lambda a{:}\phi.r) : \phi \to \psi}\,(\mathbf{L}) \qquad \frac{\Gamma \vdash r : \phi \to \psi \ \ \Gamma \vdash s : \phi}{\Gamma \vdash rs : \psi}\,(\mathbf{A})$$

Figure 1: Typing rules for the simply-typed λ-calculus (STLC)

structure. This yields a new class of models which seems to not display certain pathologies of the 'ordinary' models, and which can be leveraged to design novel calculi with applications e.g. to existential variables.

2 Background

Background on simply-typed λ-calculus

Definition 2.1. Fix a countably infinite set of **atoms** $\mathbb{A}$.

We use a **permutative** convention that $a, b, c, \ldots$ range over *distinct* atoms (so for instance in Definition 2.3 the a_i are silently assumed distinct, in Definition 2.8 a and b are taken distinct, and so on).

Definition 2.2.

1. Fix a nonempty set of **base types** $\tau \in \mathit{BaseTypes}$. Define **(simple) types** by $\phi ::= \tau \mid \phi \to \phi$. Let ϕ, ψ, χ range over types.
2. Fix a set of **constants** $C \in \mathit{Constants}$, to each of which is associated a type $type(C)$. Define **terms** by: $r ::= a \mid C \mid \lambda a{:}\phi.r \mid rr$. Let r, s, t range over terms. λa binds a in $\lambda a{:}\phi.s$ and we take terms up to α-equivalence as usual.

Define **free atoms** $fa(r)$ by $fa(a) = \{a\}$, $fa(C) = \varnothing$, $fa(rs) = fa(r) \cup fa(s)$, and $fa(\lambda a{:}\phi.r) = fa(r) \setminus \{a\}$.

Definition 2.3. Give terms a **capture-avoiding substitution action** $r[a_i{:=}s_i]_1^n$ (side-conditions can be guaranteed by α-renaming):

$$\begin{array}{rlrl}
a_j[a_i{:=}s_i]_1^n &= s_j & b[a_i{:=}s_i]_1^n &= b \\
C[a_i{:=}s_i]_1^n &= C & (\lambda c{:}\chi.r)[a_i{:=}s_i]_1^n &= \lambda c{:}\chi.(r[a_i{:=}s_i]_1^n) \quad (c \notin \bigcup_1^n(\{a_i\} \cup fa(s_i))) \\
(rs)[a_i{:=}s_i]_1^n &= (r[a_i{:=}s_i]_1^n)(s[a_i{:=}s_i]_1^n) & &
\end{array}$$

Definition 2.4. Let $=_\beta$ be the least equivalence on terms (up to α-equivalence) such that:

$$\frac{r =_\beta r' \quad s =_\beta s'}{rs =_\beta r's}\,(\mathbf{CongApp}) \qquad \frac{s =_\beta s'}{\lambda a{:}\phi.s =_\beta \lambda a{:}\phi.s'}\,(\xi) \qquad \frac{}{(\lambda a{:}\phi.r)t =_\beta r[a := t]}\,(\beta)$$

Definition 2.5. A **type environment** Γ is a set of **atomic typings** $a : \phi$ which is *functional* in the sense that if $a : \phi$ and $a : \phi'$ then $\phi = \phi'$.

Derivable typing judgements $\Gamma \vdash r : \phi$ are defined using the (standard) rules in Figure 1.

Define the **domain** of Γ by $dom(\Gamma) = \{a \mid \exists \phi.(a{:}\phi \in \Gamma)\}$. Write $\Gamma, a{:}\phi$ for the type environment obtained by adding $a{:}\phi$ to Γ; if we write this, we impose a condition that $a \notin dom(\Gamma)$.

Definition 2.6. A **typing judgement** is a tuple $\Gamma \vdash r : \phi$. The **derivable** typing judgements are defined in Figure 1.

Background on nominal sets

Definition 2.7. The cumulative hierarchy of **ZFA sets** $\mathcal{U}$ is the least fixed point of $\mathcal{U} = \mathbb{A} \cup \mathit{powerset}(\mathcal{U})$. This can be constructed by starting from atoms and transfinitely adding all subsets (a construction going back to Von Neumann [32]).

Definition 2.8. Given $a,b \in \mathbb{A}$ write $(a\ b)$ for the **swapping** bijection on atoms mapping a to b, b to a, and any other $c \in \mathbb{A} \setminus \{a,b\}$ to c.

If π is a bijection on atoms define $\mathit{nontriv}(\pi) = \{a \mid \pi(a) \neq a\}$.

Write $\mathbb{P}$ for the group of bijections (finitely) generated by swappings, and call these bijections **permutations**.

Write $\pi \circ \pi'$ for the **composition** of π and π' (so $(\pi \circ \pi')(a) = \pi(\pi'(a))$). Write id for the **identity** permutation (so $\mathit{id}(a) = a$ always).

Lemma 2.9. *A bijection π on atoms is a permutation if and only if $\mathit{nontriv}(\pi) = \{a \mid \pi(a) \neq a\}$ is finite.*

Definition 2.10. Give $\mathcal{U}$ a **permutation action** $\pi{\cdot}x$ inductively defined by $\pi{\cdot}a = \pi(a)$ for $a \in \mathbb{A}$, and $\pi{\cdot}X = \{\pi{\cdot}x \mid x \in X\}$ for $X \in \mathcal{U} \setminus \mathbb{A}$.

If $A \subseteq \mathbb{A}$ write $\mathit{fix}(A) = \{\pi \in \mathbb{P} \mid \forall a{\in}A.\pi(a) = a\}$.

Say that $A \subseteq \mathbb{A}$ **supports** $x \in \mathcal{U}$ when $\forall \pi{\in}\mathit{fix}(A).\pi{\cdot}x = x$.

Definition 2.11. Call an element $x \in \mathcal{U}$ **finitely-supported** when it has a unique least finite supporting set $\mathit{supp}(x)$. Write $a\#x$ for $a \notin \mathit{supp}(x)$ and read this as 'a is **fresh for** x'.

Lemma 2.12 ([24, 14]). *If $x \in \mathcal{U}$ has a finite supporting set A, then $\mathit{supp}(x)$ exists.*

Our reasoning can be formalised in first-order logic, enriched with the axioms of Zermelo-Fraenkel set theory with atoms (**ZFA**). This is just a formal way of stating that we have assumed atoms and sets and we reason about them mathematically, but stating it in terms of formal logic lets us express an important observation, that our reasoning is symmetric under permutation:

Theorem 2.13. *If $\overline{x}$ denotes a list $x_1,\dots,x_n$, write $\pi{\cdot}\overline{x}$ for $\pi{\cdot}x_1,\dots,\pi{\cdot}x_n$. Suppose $\Phi(\overline{x})$ is a ZFA predicate on variables included in $\overline{x}$. Then we have* equivariance *[14, Section 4]:*[1] $\Phi(\overline{x}) \Leftrightarrow \Phi(\pi{\cdot}\overline{x})$.

We will appeal to equivariance repeatedly to quickly yet rigorously rename atoms, usually while retaining an inductive hypothesis. See for instance Lemma 3.5.[2]

Definition 2.14. Say that $X \in \mathcal{U} \setminus \mathbb{A}$ has the **trivial** action when $\mathit{supp}(x) = \varnothing$ for every $x \in X$ (equivalently: when $\pi{\cdot}x = x$ for every $x \in X$ and permutation π).

Definition 2.15. If $X,Y \in \mathcal{U} \setminus \mathbb{A}$ then a **function(-set)** from X to Y is a subset of $X \times Y$ such that $\forall x{\in}X.\exists y{\in}Y.(x,y) \in f$ and $\forall x,y,y'.((x,y) \in f \wedge (x,y') \in f) \Rightarrow y = y'$. Write $X{\to}Y$ for the set of all functions from X to Y. Write $X{\Rightarrow}Y$ for the set of all functions from X to Y with finite support.

Remark 2.16. The permutation action from Definition 2.10 gives $f \in X{\to}Y$ the *conjugation* permutation action specified by $\pi{\cdot}(f(x)) = (\pi{\cdot}f)(\pi{\cdot}x)$.

Lemma 2.17. *If X and Y in $\mathcal{U} \setminus \mathbb{A}$ have the trivial permutation action (Definition 2.14), then so does $X{\to}Y$, and $X{\to}Y = X{\Rightarrow}Y$. (If underlying sets have empty support then so do functions between them.)*

[1] $\overline{x}$ must contain *all* the variables mentioned in the predicate. It is not the case that $a = a$ if and only if $a = b$—but it is the case that $a = b$ if and only if $b = a$.

[2] This technique was used in pencil-and-paper mathematics instead of long inductive proofs, e.g. in [12].

3 Nominal models for simple type theory

3.1 Nominal λ-model

Notation 3.1. Write $\pi{\cdot}\Gamma = \{\pi(a){:}\phi \mid a{:}\phi \in \Gamma\}$.

Definition 3.2. A **model** $\mathcal{I}$ consists of an assignment for each type environment Γ and type ϕ of a finitely-supported set $[\![\phi]\!]^{\mathcal{I}}_{\Gamma}$ together with the following data:

1. For every $a{:}\phi \in \Gamma$ an element $a^{\mathcal{I}}_{\phi} \in [\![\phi]\!]^{\mathcal{I}}_{\Gamma}$.
2. For every constant C an element $C^{\mathcal{I}} \in [\![\mathit{type}(C)]\!]^{\mathcal{I}}_{\Gamma}$.
3. If $x \in [\![\psi]\!]^{\mathcal{I}}_{\Gamma,a:\phi}$, an element $[a{:}\phi]x \in [\![\phi{\to}\psi]\!]^{\mathcal{I}}_{\Gamma}$.
4. For $x \in [\![\phi{\to}\psi]\!]^{\mathcal{I}}_{\Gamma}$ and $y \in [\![\phi]\!]^{\mathcal{I}}_{\Gamma}$, an element $x \bullet y \in [\![\psi]\!]^{\mathcal{I}}_{\Gamma}$.
5. $[\![\phi]\!]^{\mathcal{I}}_{\Gamma\cap\Gamma'} = [\![\phi]\!]^{\mathcal{I}}_{\Gamma} \cap [\![\phi]\!]^{\mathcal{I}}_{\Gamma'}$.
6. If $x \in [\![\phi]\!]^{\mathcal{I}}_{\Gamma}$ then $\mathit{supp}(x) \subseteq \mathit{dom}(\Gamma)$.

$\mathcal{I}$ must be *equivariant* in the sense that:

$$\pi{\cdot}[\![\phi]\!]^{\mathcal{I}}_{\Gamma} = \{\pi{\cdot}x \mid x \in [\![\phi]\!]^{\mathcal{I}}_{\Gamma}\} = [\![\phi]\!]^{\mathcal{I}}_{\pi\cdot\Gamma} \qquad \pi{\cdot}a^{\mathcal{I}}_{\phi} = (\pi(a))^{\mathcal{I}}_{\phi} \qquad \pi{\cdot}C^{\mathcal{I}} = C^{\mathcal{I}}$$
$$\pi{\cdot}[a{:}\phi]x = [\pi(a){:}\phi]\pi{\cdot}x \qquad \pi{\cdot}(x \bullet y) = (\pi{\cdot}x) \bullet (\pi{\cdot}y)$$

We write $x[a \mapsto y]$ as sugar for $([a{:}\phi]x) \bullet y$. In addition, $\mathcal{I}$ must be a **nominal algebra for substitution** by satisfying rules (**Suba**), (**Sub#**), (**SubApp**), and (**Subλ**); we fill in types as appropriate (we discuss (**SubId**) below):

$$\begin{array}{lrll}
(\textbf{Suba}) & & a^{\mathcal{I}}_{\phi}[a \mapsto x] & = x \\
(\textbf{Sub\#}) & a\#z \Rightarrow & z[a \mapsto x] & = z \\
(\textbf{SubApp}) & & (z' \bullet z)[a \mapsto x] & = (z'[a \mapsto x]) \bullet (z[a \mapsto x]) \\
(\textbf{Sub}\boldsymbol{\lambda}) & c\#x \Rightarrow & ([c{:}\chi]z)[a \mapsto x] & = [c{:}\chi](z[a \mapsto x]) \\
(\textbf{SubId}) & & z[a \mapsto a^{\mathcal{I}}] & = z
\end{array}$$

For the rest of this subsection fix a model $\mathcal{I}$.

Let us break down the design of Definition 3.2. Obviously names inhabit the denotation in a very direct and literal sense that $a^{\mathcal{I}}_{\phi} \in [\![\phi]\!]^{\mathcal{I}}_{\Gamma}$. The reader can think of $a : \phi$ as a constant which must be interpreted 'as itself' by $a^{\mathcal{I}}_{\phi}$.

But $a^{\mathcal{I}}_{\phi}$ also behaves like a variable: It can be renamed by $\pi{\cdot}x$, and bound by $[a{:}\phi]x$, and it can also be substituted for. The rules (**Suba**) to (**Subλ**) do the job that valuations do in 'normal' models; they replace a name $a^{\mathcal{I}}_{\phi}$ by an(other) element of the model. The significant difference is that in standard models we pick a valuation and then form a denotation; in nominal models we form a denotation and then—if we wish—substitute for the free variables.

The axioms (**Suba**) to (**SubId**) can be made formal in nominal algebra [22]. These particular axioms are taken from [20].[3] Instead of (**Sub#**) we could take a weaker axiom $b[a{\mapsto}x] = b$. Conversely, we could safely add (**SubId**) and thus exclude certain arguably pathological models. The language of Definition 2.2 is not expressive enough to detect these choices, but the language of Definition 4.3 is (see Example 5.10).

[3] (**Suba**) to (**SubId**) soundly and completely characterise the *syntactic* model of substitution. In this paper we are also interested in non-syntactic models, so weaker axioms—and thus more models—are reasonable. We chose the axioms above because they are *closed*, in the sense of [9, 10], which gives better computational properties (if we ever design an abstract machine using this semantics).

Aside from the inclusion of names, our notion of model resembles Henkin models, which have an applicative structure in which abstractions have a well-defined interpretation [26].

Just as is the case for Henkin models, Definition 3.2 specifies what a model must look like but does not build one. We do build a concrete model out of syntax as part of the completeness proof in Subsection 3.3.

The equivariance conditions are standard for nominal techniques; our models must be symmetric up to permuting atoms.

Finally, conditions 1 to 6 specify the structure of a model that makes it into a model of the λ-calculus, by interpreting names (as themselves), constants, λ-abstraction (as a function of the name a and the element x)[4] and a Henkin-models style application.

Definition 3.3. Suppose $\Gamma \vdash r : \phi$. Define an **interpretation** $[\![r]\!]^{\varsigma}_{\Gamma} \in [\![\phi]\!]^{\varsigma}_{\Gamma}$ inductively by:

$$\boxed{[\![a]\!]^{\varsigma}_{\Gamma,a:\phi} = a^{\varsigma}_{\phi} \qquad [\![C]\!]^{\varsigma}_{\Gamma} = C^{\varsigma} \qquad [\![rs]\!]^{\varsigma}_{\Gamma} = [\![r]\!]^{\varsigma}_{\Gamma} \bullet [\![s]\!]^{\varsigma}_{\Gamma} \qquad [\![\lambda a{:}\phi.r]\!]^{\varsigma}_{\Gamma} = [a{:}\phi][\![r]\!]^{\varsigma}_{\Gamma,a:\phi}}$$

We now come to our first soundness theorem; if a term is typable then its denotation inhabits the denotation of its type:

Theorem 3.4 (First soundness theorem). *If $\Gamma \vdash r : \phi$ then $[\![r]\!]^{\varsigma}_{\Gamma} \in [\![\phi]\!]^{\varsigma}_{\Gamma}$.*

As a corollary using condition 5 of Definition 3.2, if $\Gamma \vdash r : \phi$ then $supp([\![r]\!]^{\varsigma}_{\Gamma}) \subseteq fa(r)$.

Proof. By a straightforward induction on the derivation of $\Gamma \vdash r : \phi$:

- By the definition of model (Definition 3.2), if $a : \phi \in \Gamma$ then $[\![a]\!]^{\varsigma}_{\Gamma} = a^{\varsigma}_{\phi} \in [\![\phi]\!]^{\varsigma}_{\Gamma}$, and $[\![C]\!]^{\varsigma}_{\Gamma} = C^{\varsigma} \in [\![type(C)]\!]^{\varsigma}_{\Gamma}$.
- Suppose $\Gamma, a{:}\phi \vdash r : \psi$ so that by $(\mathbf{L})$ $\Gamma \vdash \lambda a{:}\phi.r : \phi \to \psi$ and by inductive hypothesis $[\![r]\!]^{\varsigma}_{\Gamma,a:\phi} \in [\![\psi]\!]^{\varsigma}_{\Gamma}$. By assumption $[\![\lambda a{:}\phi.r]\!]^{\varsigma}_{\Gamma} \in [\![\phi \to \psi]\!]^{\varsigma}_{\Gamma}$.
- If $[\![r]\!]^{\varsigma}_{\Gamma} \in [\![\phi \to \psi]\!]^{\varsigma}_{\Gamma}$ and $[\![s]\!]^{\varsigma}_{\Gamma} \in [\![\phi]\!]^{\varsigma}_{\Gamma}$ then $[\![r]\!]^{\varsigma}_{\Gamma} \bullet [\![s]\!]^{\varsigma}_{\Gamma} \in [\![\phi]\!]^{\varsigma}_{\Gamma}$. □

3.2 Soundness for β-conversion

Lemma 3.5. *Suppose $\Gamma, a{:}\phi \vdash r : \psi$ and $\Gamma \vdash s : \phi$, where $a \notin dom(\Gamma)$.*

Then $[\![r[a := s]]\!]^{\varsigma}_{\Gamma} = [\![r]\!]^{\varsigma}_{\Gamma,a:\phi}[a \mapsto [\![s]\!]^{\varsigma}_{\Gamma}]$.

Proof. By a routine induction on the derivation of $\Gamma, a{:}\phi \vdash r : \psi$:

- *The case of $(\mathbf{V})$ for a.* By $(\mathbf{Suba})$ $a^{\varsigma}_{\phi}[a \mapsto [\![s]\!]^{\varsigma}_{\Gamma}] = [\![s]\!]^{\varsigma}_{\Gamma}$. Also $a[a := s] = s$.
- *The case of $(\mathbf{V})$ for $c{:}\chi \in \Gamma$.* By $(\mathbf{Sub\#})$ $c^{\varsigma}_{\chi}[a \mapsto [\![s]\!]^{\varsigma}_{\Gamma}] = c^{\varsigma}_{\chi}$. Also $c[a := s] = s$.
- *The case of $(\mathbf{L})$ for $\lambda c{:}\chi.r$.* By equivariance (Theorem 2.13) suppose $c \notin dom(\Gamma) \cup supp([\![s]\!]^{\varsigma}_{\Gamma})$.[5] The result follows using $(\mathbf{Sub}\lambda)$.
- *The case of $(\mathbf{A})$* uses $(\mathbf{SubApp})$. □

Proposition 3.6 (The ξ rule). *Suppose $\Gamma, a{:}\phi \vdash r : \psi$ and $\Gamma, a{:}\phi \vdash s : \psi$.*

If $[\![r]\!]^{\varsigma}_{\Gamma,a:\phi} = [\![s]\!]^{\varsigma}_{\Gamma,a:\phi}$ then $[\![\lambda a{:}\phi.r]\!]^{\varsigma}_{\Gamma} = [\![\lambda a{:}\phi.s]\!]^{\varsigma}_{\Gamma}$.

Proof. Immediate since by Definition 3.3 $[\![\lambda a{:}\phi.r]\!]^{\varsigma}_{\Gamma} = [a{:}\phi][\![r]\!]^{\varsigma}_{\Gamma,a:\phi}$, and similarly for s. □

[4] So $[a{:}\phi]x$ need not be precisely equal to the Gabbay-Pitts atoms-abstraction $[a]x$ from [24].

[5] In fact by Theorem 3.4 $c \notin supp([\![s]\!]^{\varsigma}_{\Gamma})$ follows from $c \notin dom(\Gamma)$. But that does not matter; we can just rename c 'fresh', without having to engage in detailed calculations about how fresh it is.

Example 3.7. Proposition 3.6 does not hold in a valuation semantics of ‘ordinary’ models. For instance, consider a (valuation-based) semantics with one base type τ with denotation $\{0,1\}$ (a two-element set). Consider x and y and a valuation ρ mapping x and y both to 0. Then $[\![x]\!]_\rho = 0 = [\![y]\!]_\rho$ but $[\![\lambda x{:}\tau.x]\!]_\rho \neq [\![\lambda x{:}\tau.y]\!]_\rho$.

Corollary 3.8 (Second soundness theorem). *If $r =_\beta s$ (Defn. 2.4) and $\Gamma \vdash r : \phi$ then $[\![r]\!]^{\mathcal{I}}_\Gamma = [\![s]\!]^{\mathcal{I}}_\Gamma$.*

Proof. By some routine sets calculations, using Lemma 3.5 and Proposition 3.6. □

3.3 Completeness

Definition 3.9. Write $\mathcal{I};\Gamma \vDash r{=}s$ when there exists ϕ (which is unique if it exists) such that $\Gamma \vdash r{:}\phi$ and $\Gamma \vdash s{:}\phi$, and $[\![r]\!]^{\mathcal{I}}_\Gamma{=}[\![s]\!]^{\mathcal{I}}_\Gamma$ and $[\![r]\!]^{\mathcal{I}}_\Gamma \in [\![\phi]\!]^{\mathcal{I}}_\Gamma$. We call the **(typed) equality** $\Gamma \vdash r{=}s$ **valid** in $\mathcal{I}$.

We need one technical fact about nominal sets, for Theorem 3.11:

Lemma 3.10. *Suppose $\Gamma \vdash r : \phi$. If $a \notin supp(\langle r\rangle_\beta)$ then there exists s such that $\mathsf{T};\Gamma \vdash r = s$ and $a \notin fa(s)$.*

Proof. Using [15, Lemma 7.6.2]. □

Theorem 3.11. $\Gamma \vDash r = s$ *implies* $r =_\beta s$.

Proof. We take as our model $\mathcal{I}$ where $[\![r]\!]^{\mathcal{I}}_\Gamma = \langle r\rangle_\beta$ and $[\![\phi]\!]^{\mathcal{I}}_\Gamma = \{\langle r\rangle_\beta \mid \Gamma \vdash r : \phi\}$, and:

- If $a : \phi \in \Gamma$ then we take $a^{\mathcal{I}}_\phi = \langle a\rangle_\beta \in [\![\phi]\!]^{\mathcal{I}}_\Gamma$.
- If $\langle r\rangle_\beta \in [\![\psi]\!]^{\mathcal{I}}_{\Gamma,a:\phi}$ then we take $[a{:}\phi]\langle r\rangle_\beta = \langle\lambda a{:}\phi.r\rangle_\beta$.
- Similarly, we take $\langle r\rangle_\beta \bullet \langle s\rangle_\beta = \langle rs\rangle_\beta$.

It is a fact that $r =_\beta r'$ and $s =_\beta s'$ imply $\lambda a{:}\phi.r =_\beta \lambda a{:}\phi.r'$ and $rs =_\beta r's'$, and it follows that the definition above is well-defined.

We must also check validity of rules (**Suba**) to (**Subλ**). We consider two cases:

- *The case of* (**Suba**). It is a fact that $(\lambda a{:}\phi.a)s =_\beta s$.
- *The case of* (**Sub#**). Suppose $a\#\langle t\rangle_\beta$. By Lemma 3.10 there exists $t' =_\beta t$ such that $a \notin fa(t')$. So $t'[a := r] = t'$ and thus $\mathsf{T};\Gamma \vdash t'[a := r] = t'$. It follows that $\langle t[a := r]\rangle_\beta = \langle t\rangle_\beta$.

Furthermore, by construction if $\langle r\rangle_\beta = \langle s\rangle_\beta$ then $r =_\beta s$. □

The proof of Theorem 3.11 resembles the proof of completeness for Henkin models, with moderate changes to handle the ‘nominal’ models. Our models are not necessarily extensional (that is, we do not insist that $r = \lambda a.(ra)$ for a not free in r) whereas Henkin semantics usually are [26]; nevertheless it is reasonable to think of this as ‘Henkin semantics with names’. A survey of complete non-extensional semantics for STLC is in [2].

Theorem 3.11 is simpler than it could be; we could generalise it to completeness for arbitrary theories (i.e. we allow a set of equality axioms and prove completeness for the class of models that validate those axioms). We expect this generalisation to be an easy replay of the existing proof. We do not do this because the simpler case already illustrates the main points, and has useful features which we can now explore.

3.4 Well-pointedness

In Proposition 3.6 and Example 3.7 we saw that our nominal Henkin models have a desirable property that 'ordinary' models do not. We now come to another; to state it we need a definition:

Definition 3.12. A **homomorphism** F from $\mathcal{I}$ to $\mathcal{J}$ is a collection of functions F^{ϕ}_{Γ} mapping $[\![\phi]\!]^{\mathcal{I}}_{\Gamma}$ to $[\![\phi]\!]^{\mathcal{J}}_{\Gamma}$ which are:

- *Equivariant* in the sense that $\pi{\cdot}(F^{\phi}_{\Gamma}(x)) = F^{\phi}_{\pi\cdot\Gamma}(\pi{\cdot}x)$ (Notation 3.1).
- *Natural* in the sense that F commutes with atoms, constants, abstraction, and $\bullet$. So for example, $F^{\phi}_{\Gamma,a:\phi}(a^{\mathcal{I}}_{\phi}) = a^{\mathcal{J}}_{\phi}$ and $F^{\psi\to\phi}_{\Gamma}([a{:}\psi]x) = [a{:}\psi]F^{\phi}_{\Gamma,a:\phi}(x)$.

The notion of validity from Definition 3.9 is *local* in that it checks validity at one model. There is also a *global* notion, which checks validity at the model and all 'larger' ones:

Definition 3.13. Suppose $\Gamma \vdash r{:}\phi$ and $\Gamma \vdash s{:}\phi$. Say that $\mathcal{I};\Gamma \vDash_{\text{glo}} r = s$ when $\mathcal{J};\Gamma \vDash r = s$ in the sense of Definition 3.9 for every $\mathcal{J}$ such that there exists a homomorphism $F : \mathcal{I} \to \mathcal{J}$.

Lemma 3.14. *Suppose $\Gamma \vdash r : \phi$ and $F : \mathcal{I} \to \mathcal{J}$ is a homomorphism. Then $F^{\phi}_{\Gamma}([\![r]\!]^{\mathcal{I}}_{\Gamma}) = [\![r]\!]^{\mathcal{J}}_{\Gamma}$.*

Proof. By a routine induction on the derivation of $\Gamma \vdash r : \phi$, using naturality. □

Theorem 3.15 (Well-pointedness). *Suppose $\Gamma \vdash r : \phi$ and $\Gamma \vdash s : \phi$. Then $\mathcal{I};\Gamma \vDash r = s$ if and only if $\mathcal{I};\Gamma \vDash_{glo} r = s$.*

Proof. By considering the *identity* homomorphism from $\mathcal{I}$ to itself, mapping $x \in [\![\phi]\!]^{\mathcal{I}}_{\Gamma}$ to itself, it is clear that $\mathcal{I};\Gamma \vDash_{\text{glo}} r = s$ implies $\mathcal{I};\Gamma \vDash r = s$.

Conversely, suppose $\mathcal{I};\Gamma \vDash r = s$ and suppose F is a homomorphism from $\mathcal{I}$ to $\mathcal{J}$. By assumption $[\![r]\!]^{\mathcal{I}}_{\Gamma} = [\![s]\!]^{\mathcal{I}}_{\Gamma}$. It follows by Lemma 3.14 that $[\![r]\!]^{\mathcal{J}}_{\Gamma} = F^{\phi}_{\Gamma}([\![r]\!]^{\mathcal{I}}_{\Gamma}) = F^{\phi}_{\Gamma}([\![s]\!]^{\mathcal{I}}_{\Gamma}) = [\![s]\!]^{\mathcal{J}}_{\Gamma}$. □

Example 3.16. Theorem 3.15 fails for traditional models. For instance, consider a functional model in which all terms are equal because every type has just one element. So $\Gamma \vdash r = s$ is locally true, but not globally true.

Nominal Henkin semantics exclude this, because they have elements to interpret variables. It is impossible to compress them all down to one element, as we did in the previous paragraph for 'ordinary' models.

4 Existential variables

Nominal terms introduced to nominal techniques the idea of two levels of variable; atoms (as above) and *unknowns* X, which are existential variables and in [40] were used in a unification algorithm. The first author proposed combining nominal unknowns with non-trivial logical theories, e.g. with first-order logic [19, 21]. Since in this paper we have a nominal semantics for the STLC, it is natural to extend Definition 2.2 with nominal unknowns and so to add existential variables.

The motivation for doing this is that STLC underlies many interesting logics and programming languages, so that our semantics and syntax with existential variables have potential—not exploited in this paper but motivating the constructions—to provide syntax and semantics for 'incomplete terms'. In common with all other such treatments, a difficulty is the delicacy of maintaining well-typedness under instantiation (which for nominal terms may be capturing; see Remark 4.16). Our solution has elements of previous work, but it retains a distinct identity and remains typically 'nominal'.

We will use *permissive* nominal terms [7], which improve on the theory of α-equivalence of nominal terms by allowing us to 'just quotient' syntax (nominal terms require a freshness context and freshness context update, which are harder to manage in the presence of non-trivial equalities/reductions on terms).

4.1 Syntax

Definition 4.1. Fix a partition of the set of atoms from Definition 2.1 into two disjoint countably infinite sets $\mathbb{A}^{<}$ and $\mathbb{A}^{>}$, so that $\mathbb{A} = \mathbb{A}^{<} \uplus \mathbb{A}^{>}$

Splitting $\mathbb{A}$ in two is key to the syntax, but not to the semantics: the notion of model in Definition 5.3 is identical to Definition 3.2 and is based on finitely-supported nominal sets as usual. Only the syntax uses the more powerful notions of $\mathbb{A}^{<}$ and $\mathbb{A}^{>}$ (and it is more powerful; see e.g. Example 5.10). This echoes the formal distinction between 'names that exist to be bound' and 'names that exist to be free' used in some treatments of logic [25, 39], though this distinction is not so rigid here; e.g. a 'standalone atom' a can appear either from $\mathbb{A}^{<}$ or $\mathbb{A}^{>}$ and via a permutation or substitution 'migrate' between them.

Definition 4.2. Fix a countably infinite set of **unknowns**. X, Y, Z will range over distinct unknowns.

Definition 4.3. Types are as in Definition 2.2. Terms are defined by:

$$r ::= a \mid C \mid X[b_i{:=}s_i]_{i=1}^{n} \mid \lambda c{:}\phi.s \mid rs \qquad (\{b_i \mid 1{\leq}i{\leq}n\} \subseteq \mathbb{A}^{<},\ \ c \in \mathbb{A}^{>})$$

$[b_i := s_i]$ is a **(level 1) substitution**, which is a finite partial function from atoms to terms, mapping b_i to s_i and undefined elsewhere (so finite substitutions are directly in this syntax, just like finite permutations on unknowns $\pi{\cdot}X$ are directly part of nominal terms). We call $X[b_i{:=}s_i]_1^n$ a **moderated unknown**.

The condition $c \in \mathbb{A}^{>}$ may seem odd—so $\lambda a{:}\phi.a$ is not well-formed syntax if $a \in \mathbb{A}^{<}$—but since a is supposed to be bound we can intuitively always α-convert it to be in $\mathbb{A}^{>}$. This is a useful 'hygiene' simplification, since just by looking at an atom a we can tell if it could be bound ($a \in \mathbb{A}^{>}$) or captured by an instantiation ($a \in \mathbb{A}^{<}$). We can always move between one world and the other using a moderating substitution, as in $\lambda a{:}\phi.X[b{:=}a]$ where $a \in \mathbb{A}^{>}$ and $b \in \mathbb{A}^{<}$.

Example 4.4.

- *An incomplete term.* The typing $a,b{:}\phi,X{:}\phi \vdash \lambda a{:}\phi.X[b{:=}a] : \phi{\to}\phi$ where $a \in \mathbb{A}^{>}$ and $b \in \mathbb{A}^{<}$ represents an incomplete typing '$\lambda x{:}\phi.t$ where t has type ϕ'. This is an term for a function on one argument.

  Looking forward to the level 1 and 2 substitutions in Definitions 4.10 and 5.1, we will be able to complete $\lambda a{:}\phi.X$ to a complete term, by applying the substitution $[X := b]$. We get the identity $\lambda a{:}\phi.a$. Without unknowns, both the incomplete and the complete terms would be represented by $\lambda a{:}\phi.fa$ for a higher-order $f : \phi{\to}\phi$ (whereas X has type ϕ).
- *An incomplete HOL predicate.* Assume base types ι and o and constants $\dot{\Rightarrow} : o{\to}o{\to}o$ and $\dot{\forall} : (\iota{\to}o){\to}o$. The typing $X : o,\ Y : \iota,\ b{:}\iota \vdash (\dot{\forall}\lambda b{:}\iota.X)\dot{\Rightarrow}X[b := Y] : o$ represents an incomplete HOL predicate.

  Without level 2 variables, both the incomplete and the complete terms would be represented by $b{:}\iota,\ f : \iota{\to}o \vdash (\dot{\forall}\lambda b{:}\iota.fa)\dot{\Rightarrow}fa$.

Definition 4.5. Suppose a permutation π (Definition 2.8) is such that $\mathit{nontriv}(\pi) \subseteq \mathbb{A}^{>}$. Define a **permutation action** $\pi{\cdot}r$ on terms by:

$$\begin{array}{rcc} \pi{\cdot}a = \pi(a) & \pi{\cdot}\lambda a{:}\phi.r = \lambda\pi(a){:}\phi.\pi{\cdot}r & \pi{\cdot}C = C \\ \pi{\cdot}(rs) = (\pi{\cdot}r)(\pi{\cdot}s) & \pi{\cdot}(X\,[b_i := s_i]_i) = X\,[b_i := \pi{\cdot}s_i]_i & \end{array}$$

Remark 4.6. Intuitively, the reason that we restrict $\mathit{nontriv}(\pi)$ to atoms in $\mathbb{A}^{>}$ is so that we only rename the atoms that can be λ-abstracted. This restriction could be removed, and the syntax made 'more equivariant', but at the price of complicating the syntax $X[b_i := s_i]_1^n$ to $(\pi{\cdot}X)[b_i := s_i]_1^n$ so that we could write $\pi{\cdot}(X\,[b_i := s_i]_i = (\pi{\cdot}X)[\pi(b_i){:=}\pi{\cdot}s_i]$. There would be nothing wrong with this—it just makes our basic syntax slightly more complicated. Since there is no change in expressivity, we leave this out.

We could also emulate π using the substitution $[b_i := s_i]_1^n$, but then we must add (**SubId**).

Definition 4.7. Call a binary relation $\mathscr{R}$ on terms a **congruence** when it is closed under the rules (**CongApp**) and (ξ) in Definition 2.4 and in addition:[6]

$$\frac{s_i \mathrel{\mathscr{R}} s_i' \quad (1{\leq}i{\leq}n,\ \{b_1,\dots,b_n\} \subseteq \mathbb{A}^{<})}{X\,[b_i{:=}s_i]_1^n \mathrel{\mathscr{R}} X\,[b_i{:=}s_i']_1^n}\ (\mathbf{CongX})$$

Definition 4.8. α-equivalence $r =_\alpha s$ is the least congruence such that if $b \in \mathbb{A}^{>} \setminus \mathit{fa}(r)$ then $\lambda a{:}\phi.r =_\alpha \lambda b{:}\phi.(b\ a){\cdot}r$. Henceforth we quotient terms by α-equivalence.

Definition 4.9. Define **free atoms** $\mathit{fa}(r)$ and **free unknowns** $\mathit{fv}(r)$ by:

$$\begin{array}{rcc} \mathit{fa}(a) = \{a\} & \mathit{fa}(\lambda a{:}\phi.r) = \mathit{fa}(r) \setminus \{a\} & \mathit{fa}(C) = \varnothing \\ \mathit{fa}(rs) = \mathit{fa}(r) \cup \mathit{fa}(s) & \mathit{fa}(X[b_i{:=}s_i]_1^n) = (\mathbb{A}^{<} \setminus \{b_i \mid i\}) \cup \bigcup_i \mathit{fa}(s_i) & \end{array}$$

$$\begin{array}{rcc} \mathit{fv}(a) = \varnothing & \mathit{fv}(\lambda a{:}r.s) = \mathit{fv}(s) \cup \mathit{fv}(r) & \mathit{fv}(C) = \varnothing \\ \mathit{fv}(rs) = \mathit{fv}(r) \cup \mathit{fv}(s) & \mathit{fv}(X[a_i{:=}s_i]_1^n) = \{X\} \cup \bigcup_i \mathit{fv}(s_i) & \end{array}$$

Definition 4.10. We give terms a capture-avoiding **substitution action** $r[b_i := s_i]_1^n$ as follows:

$$\begin{array}{rcll} b_j[b_i{:=}s_i]_1^n &=& s_j & (1{\leq}j{\leq}n) \\ a[b_i{:=}s_i]_1^n &=& a & (a \notin \{b_i \mid 1{\leq}i{\leq}n\}) \\ C[b_i{:=}s_i]_1^n &=& C & \\ X\,[b_i{:=}s_i]_{i\in A}[b_i{:=}s_i]_{i\in B} &=& X\,[\,(b_i{:=}s_i)_{i\in B\setminus A,\ b_i\in\mathbb{A}^{<}}\,, & (B = \{1,\dots,n\}) \\ && \quad (b_i{:=}s_i[b_j{:=}s_j]_{j\in B})_{i\in A}] & \\ (\lambda c{:}\phi.r)[b_i{:=}s_i]_1^n &=& \lambda c{:}\phi.(r[b_i{:=}s_i]_1^n) & (c \in \mathbb{A}^{>} \setminus (\bigcup_i \{b_i\} \cup \mathit{fa}(s_i))) \\ (r'r)[b_i{:=}s_i]_1^n &=& (r'[b_i{:=}s_i]_1^n)(r[b_i{:=}s_i]_1^n) & \end{array}$$

Note above that if $b_i \in \mathbb{A}^{>}$ then it gets garbage-collected (eliminated) on X, as we see from the condition '$b_i \in \mathbb{A}^{<}$' in '$i \in B\setminus A,\ b_i \in \mathbb{A}^{<}$'. So for instance $X\,[b{:=}b'][b'{:=}b''] = X\,[b{:=}b'', b'{:=}b'']$ where $b, b', b'' \in \mathbb{A}^{<}$ and $X\,[b{:=}a][a{:=}b''] = X\,[b{:=}b'']$ where $a \in \mathbb{A}^{>}$.

Definition 4.11. Let **β-equivalence** $- =_\beta -$ be the least congruence (Definition 4.7) such that $(\lambda a{:}\phi.r)t =_\beta r[a := t]$.

[6] The condition $\{b_1,\dots,b_n\} \subseteq \mathbb{A}^{<}$ is there to guarantee that $X\,[b_i{:=}s_i]$ and $X\,[b_i{:=}s_i']$ are well-formed terms.

$$\frac{(a:\phi\in\Gamma)}{\Gamma\vdash a:\phi}\,(\mathbf{V})\qquad\frac{(\mathit{type}(C)=\phi)}{\Gamma\vdash C:\phi}\,(\mathbf{C})\qquad\frac{\Gamma,a{:}\phi\vdash r:\psi\quad(a\in\mathbb{A}^{>})}{\Gamma\vdash(\lambda a{:}\phi.r):\phi\to\psi}\,(\mathbf{L})$$

$$\frac{\Gamma\vdash r:\phi\to\psi\quad\Gamma\vdash s:\phi}{\Gamma\vdash rs:\psi}\,(\mathbf{A})\qquad\frac{\Gamma\vdash s_i:\psi_i\quad(X{:}\phi{\in}\Gamma,\ b_i{:}\psi_i{\in}\Gamma,\ 1{\leq}i{\leq}n)}{\Gamma\vdash X\,[b_i{:=}s_i]_1^n:\phi}\,(\mathbf{Meta})$$

Figure 2: Typing rules for the simply-typed λ-calculus with holes

4.2 Environments and typing

Definition 4.12. A **type environment** Γ is a set of **atomic typings** $a:\phi$ or $X:\phi$ which is *functional* in the sense that if $a:\phi$ and $a:\phi'$ then $\phi=\phi'$, and similarly for X (i.e. 'add $X{:}\phi$ to Definition 2.5').

Define $\mathit{dom}(\Gamma)=\{a\mid\exists\phi.a{:}\phi\in\Gamma\}\cup\{X\mid\exists\phi.X{:}\phi\in\Gamma\}$.

Definition 4.13. Define a **typing relation** by the rules in Figure 2.

One interesting feature of Figure 2 is that b_i *must* be typed in Γ in $(\mathbf{Meta})$. This means that we can strengthen only for atoms in $\mathbb{A}^{>}$ (the 'abstractable' atoms); see Lemma 4.15. See Remarks 4.16 and 4.19 for discussions of why. Also, the s_i are typed in a context in which the b_i occur. There is no problem with circularities; in the models the b_i are just elements (with special properties).

Lemma 4.14 (Weakening)**.** *If* $\Gamma\vdash r:\psi$ *and* $c\notin\mathit{dom}(\Gamma)$ *then* $\Gamma,c{:}\chi\vdash r:\psi$.

Proof. By induction on the derivation of $\Gamma\vdash r:\psi$. For the case of $(\mathbf{L})$ we may rename using equivariance (Theorem 2.13). □

Lemma 4.15 (Strengthening)**.** *If* $\Gamma,c{:}\chi\vdash r:\psi$ *and* $c\in\mathbb{A}^{>}\setminus\mathit{fa}(r)$ *then* $\Gamma\vdash r:\psi$.

Proof. By induction on the derivation of $\Gamma\vdash r:\psi$. For the case of $(\mathbf{L})$ we may rename using equivariance (Theorem 2.13). The rule $(\mathbf{Meta})$ is why we insist on $c{\notin}\mathbb{A}^{<}$; the atoms $\{b_i\mid 1{\leq}i{\leq}n\}\subseteq\mathbb{A}^{<}$ may not feature in $\mathit{fa}(r)$ but *must* be in Γ, as discussed in Remark 4.16. □

Remark 4.16. $(\mathbf{Meta})$ states that if $X{:}\phi\in\Gamma$ and $b_i{:}\psi_i\in\Gamma$ for $1{\leq}i{\leq}n$ then $\Gamma\vdash s_i:\psi_i$ for $1{\leq}i{\leq}n$ implies $\Gamma\vdash X[b_i{:=}s_i]:\phi$. We must insist on $b_i{:}\psi_i\in\Gamma$, for suppose otherwise: Then for $a\in\mathbb{A}^{>}$ and $b\in\mathbb{A}^{<}$ we could derive $X{:}\phi,a{:}\chi\vdash\lambda b{:}\phi.X[a:=b]:\phi\to\phi$; the types of a and b are inconsistent.

Lemma 4.17. $\mathit{fa}(r[a:=t])\subseteq(\mathit{fa}(r)\setminus\{a\})\cup\mathit{fa}(t)$

Lemma 4.18. $\Gamma,(c_i{:}\chi_i)_1^n\vdash r:\phi$ *and* $\Gamma,(c_i{:}\chi_i)_1^n\vdash s_j:\chi_j$ *for* $1{\leq}j{\leq}n$ *imply* $\Gamma,(c_i{:}\chi_i)_1^n\vdash r[c_i:=s_i]_1^n:\phi$.

As a corollary, if $c\in\mathbb{A}^{>}\setminus\mathit{fa}(s)$ *then* $\Gamma,c{:}\chi\vdash r:\phi$ *and* $\Gamma\vdash s:\chi$ *imply* $\Gamma\vdash r[c:=s]:\phi$.

Proof. By a routine induction on the derivation of $\Gamma,(c_i{:}\chi_i)_1^n\vdash r:\phi$. For the case of $(\mathbf{L})$ we may rename the bound atom in the derivation using equivariance (Theorem 2.13).

The corollary follows by Lemmas 4.14, 4.15, and 4.17. □

Remark 4.19. Lemma 4.18 does not state $\Gamma,c{:}\chi\vdash r:\phi$ and $\Gamma\vdash s:\chi$ imply $\Gamma\vdash r[c:=s]:\phi$, for $c\in\mathbb{A}^{<}$. For instance $X{:}\phi,c{:}\phi\vdash X:\phi$ and $X{:}\phi\vdash X:\phi$ but it is not the case that $X{:}\phi\vdash X[c:=X]:\phi$.

5 Level 2 substitution

Definition 5.1. A **level 2 substitution** is a map θ from unknowns to terms.[7] Write $[X := t]$ for the substitution mapping X to t and all other Y to Y.

$$\begin{array}{cccc} a\theta = a & (\lambda a{:}\phi.s)\theta = \lambda a{:}\phi.(s\theta) & (a \in \mathbb{A}^{>} \setminus \bigcup_{X\in fv(t)} fa(\theta(X))) & C\theta = C \\ (rs)\theta = (r\theta)(s\theta) & X[b_i{:=}s_i]_1^n\theta = \theta(X)[b_i{:=}s_i\theta]_1^n & & \end{array}$$

Proposition 5.2. *If $\Gamma, X{:}\xi \vdash r{:}\phi$ and $\Gamma \vdash t{:}\xi$ then $\Gamma \vdash r[X := t]{:}\phi$.*

Proof. By induction on the derivation of $\Gamma, X{:}\xi \vdash r{:}\phi$. The cases of $(\mathbf{V})$ and $(\mathbf{A})$ are routine:

- *The case of* $(\mathbf{Meta})$ *for* X. Suppose $\Gamma, X{:}\xi, (b_i{:}\psi_i)_1^n \vdash X[b_i{:=}s_i]_1^n : \xi$ because $\Gamma, X{:}\xi, (b_i{:}\psi_i)_1^n \vdash s_j : \psi_j$ for $1{\leq}j{\leq}n$.
  Suppose $\Gamma, (b_i{:}\psi_i)_1^n \vdash t : \xi$. By inductive hypothesis $\Gamma, (b_i{:}\psi_i)_1^n \vdash s_j[X{:=}t] : \psi_j$ for $1{\leq}j{\leq}n$. By definition $X[b_i{:=}s_i]_1^n[X{:=}t] = t[b_i{:=}s_i[X{:=}t]]_1^n$. So it suffices to show that $\Gamma, (b_i{:}\psi_i)_1^n \vdash t[b_i{:=}s_i[X{:=}t]]_1^n : \xi$. We use Lemma 4.18.
- *The case of* $(\mathbf{L})$. Renaming if necessary using equivariance (Theorem 2.13), assume $a \in \mathbb{A}^{>} \setminus fa(t)$. By definition $(\lambda a{:}\phi.r)[X{:=}t] = \lambda a{:}\phi.(r[X{:=}t])$. We use the inductive hypothesis for $\Gamma, X{:}\xi, a{:}\phi \vdash r : \psi$. □

5.1 Models

Definition 5.3. A **model** $\mathcal{J}$ consists of an assignment for each type environment Γ and type ϕ of a finitely-supported set $[\![\phi]\!]^{\mathcal{J}}_{\Gamma}$ together with the same data as in Definition 3.2, satisfying the same equivariance conditions and $(\mathbf{Suba})$ to $(\mathbf{Sub}\boldsymbol{\lambda})$ except that in addition:

8. If $\Gamma(a) = \Gamma'(a)$ for every $a \in \mathbb{A}$ then $[\![\phi]\!]^{\mathcal{J}}_{\Gamma} = [\![\phi]\!]^{\mathcal{J}}_{\Gamma'}$ (so the model ignores $X{:}\phi \in \Gamma$ and only looks at the typing of atoms).

Definition 5.4 (Simultaneous substitution). Suppose $b_i{:}\psi_i \in \Gamma$ and $y_i \in [\![\psi_i]\!]^{\mathcal{J}}_{\Gamma}$ for $1{\leq}i{\leq}n$. Suppose $x \in [\![\phi]\!]^{\mathcal{J}}_{\Gamma}$. Specify $x[b_i{\mapsto}y_i]_1^n$ to be equal to $\big(((c_1\ b_1)\circ\cdots\circ(c_n\ b_n))\cdot x\big)[c_1{\mapsto}y_1]\ldots[c_n{\mapsto}y_n]$ for fresh $c_1,\ldots,c_n$ (so $c_i \notin supp(x) \cup \bigcup_i supp(y_i)$ for $1 \leq i \leq n$).[8]

Lemma 5.5. *If $x \in [\![\phi]\!]^{\mathcal{J}}_{\Gamma}$ then $\pi{\cdot}x \in [\![\phi]\!]^{\mathcal{J}}_{\pi\cdot\Gamma}$.*

Proof. Direct from equivariance (Theorem 2.13). □

By Lemma 5.5 syntax is equivariant for atoms in $\mathbb{A}^{>}$, because the predicate we use to define it in Definition 4.3 uses a partition $\mathbb{A} = \mathbb{A}^{<} \cup \mathbb{A}^{>}$. The notion of model of Definition 5.3 does not use this partition however, so it is equivariant for *all* π, and not just those with $nontriv(\pi) \subseteq \mathbb{A}^{>}$. Lemma 5.5 depends on this, and Lemma 5.6 uses it.

Lemma 5.6. *Suppose $b_i{:}\psi \in \Gamma$ and $y_i \in [\![\psi]\!]^{\mathcal{J}}_{\Gamma}$ for $1{\leq}i{\leq}n$, and suppose $z \in [\![\chi]\!]^{\mathcal{J}}_{\Gamma}$. Then $z[b_i{\mapsto}y_i]_1^n \in [\![\chi]\!]^{\mathcal{J}}_{\Gamma}$.*

[7] The reader familiar with nominal techniques might expect a condition that $fa(\theta(X)) \subseteq \mathbb{A}^{<}$ always. This would be necessary if moderations were permutations, but is not if they are substitutions. See [15, Proposition 3.4.3].

[8] Definition 5.3 only provides substitution for one atom at a time. We need simultaneous substitution in the semantics to give meaning to level 2 variables (see Definition 5.9). The minor difficulty is that it might be that $b_i \in supp(y_j)$. So we 'rename atoms fresh' first, and then substitute for these atoms one at a time. Certain detailed but routine verifications are necessary to make sure this works and is well-defined (depends neither on the fresh choice of c_i, nor on the order in which the substitutions are then carried out). The relevant maths is described in [14, Section 6].

Proof. Unpack Definition 5.4 and use Lemma 5.5 and conditions 2 and 3 of Definition 5.3. □

Definition 5.7. A **valuation** ς is a function on unknowns such that $supp(\varsigma(X)) \subseteq \mathbb{A}^{<}$ for every X. Write $\Gamma \vDash \varsigma$ when $X{:}\phi \in \Gamma$ implies $\varsigma(X) \in [\![\phi]\!]^{\mathcal{I}}_{\Gamma}$ for every unknown X.

Remark 5.8. Definition 5.7 seems harmless, but it carries some real meaning. By condition 6 of Definitions 3.2 and 5.3, if $x \in [\![\phi]\!]^{\mathcal{I}}_{\Gamma}$ then $supp(x) \subseteq dom(\Gamma)$. This implies that if $X{:}\phi \in \Gamma$ then X ranges over elements *with support in* $dom(\Gamma)$. What happens to all the atoms in $\mathbb{A}^{<} \setminus dom(\Gamma)$? They cannot be used (unless we weaken the context with more typings).

This is related to a celebrated topic of continuing debate in the philosophy of language that assertions like 'the King of France is bald' name and assert properties of apparently non-existent objects; they have meaning but do not denote [36]. In the same way, the variable X asserts a property of all atoms in $\mathbb{A}^{<}$—that they may appear in the denotation of X—but this does not imply that these atoms exist in the possible world determined by the typing Γ. The typing context determines which of the atoms in $\mathbb{A}^{<}$ have *existential import* [29]. The extra twist to this story here, is that in nominal techniques atoms name themselves.

This is another way of looking at the fine detail of the rule (**Meta**), that $b_i{:}\psi_i \in \Gamma$ even though $b_i \notin fa(X[b_i{:=}s_i]^n_1])$ in general. In order to be substituted for, the atom b_i must exist, and to exist it must be typed.

Definition 5.9. Suppose $\Gamma \vDash \varsigma$ and $\Gamma \vdash r : \phi$. Define an **interpretation function** mapping r to $[\![r]\!]^{\mathcal{I}}_{\varsigma;\Gamma}$, by induction on r:

$$\begin{array}{rcll}
[\![a]\!]^{\mathcal{I}}_{\varsigma;\Gamma} &=& a^{\mathcal{I}}_{\phi} & (a{:}\phi \in \Gamma) \qquad [\![C]\!]^{\mathcal{I}}_{\varsigma;\Gamma} = C^{\mathcal{I}} \\
[\![\lambda a{:}\phi.r]\!]^{\mathcal{I}}_{\varsigma;\Gamma} &=& [a{:}\phi][\![r]\!]^{\mathcal{I}}_{\varsigma;\Gamma} & (a \in \mathbb{A}^{>} \setminus dom(\Gamma)) \qquad [\![rs]\!]^{\mathcal{I}}_{\varsigma} = [\![r]\!]^{\mathcal{I}}_{\varsigma;\Gamma} \bullet [\![s]\!]^{\mathcal{I}}_{\varsigma;\Gamma} \\
[\![X\,[b_i := s_i]_i]\!]^{\mathcal{I}}_{\varsigma;\Gamma} &=& \varsigma(X)\,[b_i \mapsto [\![s_i]\!]^{\mathcal{I}}_{\varsigma;\Gamma}]_i & (X{:}\phi \in \Gamma)
\end{array}$$

A few brief words on the case of $\lambda a{:}\phi.r$: The condition $a \notin dom(\Gamma)$ prevents $a{:}\phi$ from overwriting typing information in Γ. The condition $a \notin \mathbb{A}^{<}$ ensures that the clause is well-defined, since otherwise a might 'accidentally capture' an atom in $\varsigma(X)$ for $X \in fv(r)$. The effect of a capturing an atom in X can be attained e.g. as $\lambda a{:}\phi.(X[a'{:=}a])$ where $a' \in \mathbb{A}^{<}$.

The language with holes (Definition 4.3) is more expressive than the language without it (Definition 2.2). For instance, $a{:}\phi \vDash r[a{\mapsto}a] = r$ (Definition 3.9) is true for r without unknowns, but otherwise may be false. This is because without unknowns, we can use (**Suba**) to (**Subλ**) to push substitution down to the atoms until it either vanishes or substitutes. With unknowns this cannot be done; we may get 'stuck' on a moderated unknown.

Put another way, X really does range over arbitrary elements of the model whereas a can only be *substituted* for an arbitrary element of the model—and these are two distinct concepts.

Example 5.10. Consider one base type and no constants and a nominal model $\mathcal{I}$ such that $[\![\tau]\!]^{\mathcal{I}}_{a:\tau} = \{a^{\mathcal{I}}_{\tau}, 0, 1\}$, where $supp(0) = supp(1) = \varnothing$. Set $a^{\mathcal{I}}_{\tau}[a{\mapsto}x] = x$, $0[a{\mapsto}x] = 0$, and $1[a{\mapsto}x] = 0$.

In STLC we cannot detect the element 1 and its sensitivity to $[a{\mapsto}x]$ even though $a \notin supp(1)$. In STLC extended with unknowns, we can. Thus, we use (**Sub#**) instead of a weaker axiom that $b[a{\mapsto}x] = x$.

Theorem 5.11 (First soundness theorem). *If* $\Gamma \vDash \varsigma$ *and* $\Gamma \vdash r : \phi$ *then* $[\![r]\!]^{\mathcal{I}}_{\varsigma;\Gamma} \in [\![\phi]\!]^{\mathcal{I}}_{\Gamma}$.

Proof. We consider two cases; the rest is as proof of Theorem 3.4:

- *The case of* (**L**). Suppose $\Gamma, a{:}\phi \vdash r : \psi$ and $a \in \mathbb{A}^{>}$ so that by (**L**) $\Gamma \vdash \lambda a{:}\phi.r : \phi{\to}\psi$. By inductive hypothesis $\llbracket r \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma,a:\phi} \in \llbracket \psi \rrbracket^{\mathcal{I}}_{\Gamma,a:\phi}$. It follows from condition 3 of Definition 5.3 that $[a{:}\phi]\llbracket r \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma,a:\phi} \in \llbracket \phi{\to}\psi \rrbracket^{\mathcal{I}}_{\Gamma}$. By Definition 5.9, $\llbracket \lambda a{:}\phi.r \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma} \in \llbracket \phi{\to}\psi \rrbracket^{\mathcal{I}}_{\Gamma}$.
- *The case of* (**Meta**). Suppose $\Gamma \vdash s_i : \psi_i$ and $X{:}\phi \in \Gamma$ and $b_i{:}\psi_i \in \Gamma$ for $1{\leq}i{\leq}n$, so that by (**Meta**) $\Gamma \vdash X\,[b_i{:=}s_i]_i : \phi$.
  By inductive hypothesis $\llbracket s_i \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma} \in \llbracket \psi_i \rrbracket^{\mathcal{I}}_{\Gamma}$ and by assumption $\varsigma(X) \in \llbracket \phi \rrbracket^{\mathcal{I}}_{\Gamma}$. It follows by Lemma 5.6 that $\varsigma(X)\,[b_i{:=}\llbracket s_i \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}]_i \in \llbracket \phi \rrbracket^{\mathcal{I}}_{\Gamma}$. □

5.2 Soundness for β-conversion

Lemma 5.12. *Suppose $a_i{:}\phi_i \in \Gamma$ for $i \in A$ and $\Gamma \vdash r : \psi$ and $\Gamma \vdash s_i : \phi_i$ for $i \in A$. Suppose $\Gamma \vDash \varsigma$. Then $\llbracket r[a_i := s_i]_{i\in A} \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma} = \llbracket r \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}[a_i \mapsto \llbracket s_i \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}]_{i\in A}$.*

Proof. By a routine induction on the derivation of $\Gamma \vdash r : \psi$. We consider two cases:

- *The case of* (**L**) *for* $\lambda c{:}\chi.r$. Renaming if necessary, suppose c is fresh (so that $c \notin \bigcup_{i\in A}(\mathit{supp}(\llbracket s_i \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}) \cup \mathit{fa}(s_i)) \cup \mathit{dom}(\Gamma)$). We reason as follows:

$$\begin{aligned}
\llbracket \lambda c{:}\chi.r \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}[a_i \mapsto \llbracket s_i \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}]_{i\in A} &= ([c{:}\chi]\llbracket r \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma})[a_i \mapsto \llbracket s_i \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}]_{i\in A} && \text{Definition 5.9} \\
&= [c{:}\chi](\llbracket r \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}[a_i \mapsto \llbracket s_i \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}]_{i\in A}) && (\mathbf{Sub}\lambda),\ c \notin \mathit{supp}(\llbracket s \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}) \\
&= [c{:}\chi]\llbracket r[a_i := s_i]_{i\in A} \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma} && \text{ind. hyp.} \\
&= \llbracket \lambda c{:}\chi.(r[a_i := s_i]_{i\in A}) \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma} && \text{Definition 5.9} \\
&= \llbracket (\lambda c{:}\chi.r)[a_i := s_i]_{i\in A} \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma} && c \notin \mathit{fa}(s)
\end{aligned}$$

- *The case of* (**Meta**). We reason as follows, where $B = \{1,\dots,n\}$ and $\{b_j \mid j \in B\} \subseteq \mathbb{A}^{<}$:

$$\begin{aligned}
\llbracket X[a_j{:=}t_j]_{j\in B} \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}[a_i{\mapsto}\llbracket s_i \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}]_{i\in A} &= \varsigma(X)[a_j{\mapsto}\llbracket t_j \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}]_{i\in B}[a_i{\mapsto}\llbracket s_i \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}]_{i\in A} && \text{Defn. 5.9} \\
&= \varsigma(X)[(a_i{\mapsto}\llbracket s_i \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma})_{i\in A\setminus B,\ a_i\in\mathbb{A}^{<}}, (a_j{\mapsto}\llbracket t_j \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}[a_i{\mapsto}\llbracket s_i \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}]_{i\in A})_{j\in B}] && \text{fact} \\
&= \varsigma(X)[(a_i{\mapsto}\llbracket s_i \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma})_{i\in A\setminus B,\ a_i\in\mathbb{A}^{<}}, (a_j{\mapsto}\llbracket t_j[a_i{:=}s_i]_{i\in A} \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma})_{j\in B}] && \text{ind. hyp.} \\
&= \llbracket X[(a_i{:=}s_i)_{i\in A\setminus B,\ a_i\in\mathbb{A}^{<}}, (a_j{:=}t_j[a_i{:=}s_i]_{i\in A})_{j\in B}] \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma} && \text{Defn. 5.9} \\
&= \llbracket X[a_j{:=}t_j]_{j\in B}[a_i{:=}s_i]_{i\in A} \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma,a:\phi} && \text{Defn. 4.10}
\end{aligned}$$

Some detailed calculations are hidden in the 'fact' used above. This follows using (**Subλ**) and (**SubApp**) from Definition 5.4, and is one reason that in that definition we 'freshened' the b_i to c_i; to avoid clash. □

Lemma 5.13. *Write '$\Gamma \vdash r,s : \phi$' as shorthand for '$\Gamma \vdash r : \phi$ and $\Gamma \vdash s : \phi$'. Suppose $\Gamma \vDash \varsigma$.*

1. *Suppose $\Gamma \vdash r,r' : \phi{\to}\phi'$ and $\Gamma \vdash s,s' : \phi$. If $\llbracket r \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}{=}\llbracket r' \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}$ and $\llbracket s \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}{=}\llbracket s' \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}$ then $\llbracket rs \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}{=}\llbracket r's' \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}$.*
2. *Suppose $\Gamma, a{:}\phi \vdash r,r' : \psi$. If $\llbracket r \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma,a:\phi} = \llbracket r' \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma,a:\phi}$ then $\llbracket \lambda a{:}\phi.r \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma} = \llbracket \lambda a{:}\phi.r' \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}$.*
3. *Suppose $X{:}\phi \in \Gamma$ and $\{b_i{:}\phi_i \mid 1{\leq}i{\leq}n\} \subseteq \Gamma$. Suppose $\Gamma \vdash s_j, s'_j : \psi_j$ and $\llbracket s_j \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma} = \llbracket s'_j \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}$ for $1{\leq}j{\leq}n$. Then $\llbracket X[b_i{:=}s_i]^n_1 \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma} = \llbracket X[b_i{:=}s'_i]^n_1 \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}$.*

Proof. These are facts of equality in sets. □

Corollary 5.14 (Second soundness theorem). *If $r =_\beta s$ (Defn. 4.11) and $\Gamma \vdash r : \phi$ and $\Gamma \models \zeta$ then $\llbracket r \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma} = \llbracket s \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}$.*

Proof. Using Lemmas 5.12 and 5.13. □

We would expect a completeness result like Theorem 3.11 to hold and have a similar proof. Details will be in a journal version. We should verify soundness under instantiating unknowns:

Theorem 5.15 (Third soundness theorem). *Suppose* $\Gamma, X{:}\chi \vdash r : \phi$ *and* $\Gamma \vdash t : \psi$. *Suppose* $\Gamma \vDash \varsigma$. *Then* $\llbracket r[X{:=}t] \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma} = \llbracket r \rrbracket^{\mathcal{I}}_{\varsigma[X:=\llbracket t \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}];\Gamma}$.

Proof. We consider (**Meta**) for X. Suppose $\Gamma = \Gamma', X{:}\chi, (b_j{:}\psi_j)_1^n$ and $\Gamma \vdash s_i{:}\psi_i$ for $1 {\leq} i {\leq} n$ so that by (**Meta**) $\Gamma', X{:}\chi, (b_j{:}\psi_j)_1^n \vdash X\,[b_i{:=}s_i]_i$. Then $\llbracket X\,[b_i{:=}s_i]_i \rrbracket^{\mathcal{I}}_{\varsigma[X\,:=\,\llbracket t \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}];\Gamma} \overset{\text{Defn. 5.9}}{=} \llbracket t \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}[b_i {\mapsto} \llbracket s_i \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}]_i \overset{\text{Lemm. 5.12}}{=} \llbracket t\,[b_i{:=}s_i]_i \rrbracket^{\mathcal{I}}_{\varsigma;\Gamma}$. □

6 Conclusions

We have built a semantics for the simply-typed λ-calculus (STLC), based on nominal sets. In keeping with the 'nominal' philosophy, variables (names) are denoted by themselves. This draws certain structure that is normally external to the denotation inside it, and this extra structure excludes some arguably pathological homomorphisms between models. We also exploit the semantics to existential variables, or 'holes' (suggesting that we do not just get more models out of this nominal semantics, but also more languages).

The constructions are not really any harder than for traditional STLC semantics. When reading for instance Definition 3.2, the reader should mentally place this side-by-side with a *full* specification of traditional STLC semantics, including for instance a precise definition of valuations as graphs (these are functions with the general shape $(\mathbb{A} \to X) \to X$). Our nominal semantics for STLC is no harder than what the reader already knows; it is just different.

Atypically for nominal techniques so far as exemplified e.g. by [40, 38, 3, 5], atoms have non-trivial types. These, if they assign atoms any type information at all, assign them 'the type of atoms'. There has been some work assigning more interesting types to atoms [8], but not in denotations.

In the course of doing all these things, we note echoes of other strands of research. The distinction between $b \in \mathbb{A}^{<}$ and $b : \phi \in \Gamma$ is an instance of the distinction between meaning and denotation (only finitely many of the atoms in the permission set of an unknown have existential import in the denotation) [36, 29]. Our use of $\mathbb{A}^{<}$ and $\mathbb{A}^{>}$, which is borrowed from [6, 7], is reminiscent of the two kinds of variable used by Frege [25] (for a more modern presentation see e.g. [39, Chapter IV, Section 1]). This is more an analogy than a precise correspondence and we will discuss matters further in a longer paper where we have more space to develop the syntax. We still have only one set of atoms and the 'nominal' constructions, notably the notions of support, binding, freshness, and nominal set, are unchanged. We have freely imported ideas from (permissive) nominal terms, notably in our treatment of existential variables.

6.1 Related work

Valuations and unknowns. We gave unknowns a semantics using valuations in Definition 5.9. Arguably it is disappointing: why map atoms to themselves in a denotation but then switch to another (more traditional) methodology for unknowns? One answer is that atoms are universal variables (could be replaced by anything) whereas unknowns are existential variables (must be replaced by something), so it is reasonable to interpret them using a valuation, and perhaps we should. Indeed there is a precedent for this: atoms correspond to δ-*variables* and unknowns to

γ-variables from [42]. Making this formal by considering a paper similar to this one but aimed at first-order logic is a topic of current research.

Still, there is an interesting alternative. In [16] a direct nominal semantics is explored for unknowns X, analogous to how atoms a map to themselves in this paper, called *two-level nominal sets*. Two-level nominal sets *with substitutions* would provide a theory of incompleteness in which holes are directly represented in the semantics. There is no need for that in this paper because we have no level 3 variables (in the style e.g. of the λ-context calculus [18]); but if there were, two-level nominal sets might be not only interesting, but necessary.

Models as presheafs. The reader familiar with category theory will recognise in Definition 3.2 a presheaf. In fact, we have enriched the usual presheaf $\mathit{Sets}^{\mathbb{I}}$ (presheaves over finite sets and injections between them) to a presheaf over an indexing category enriched with types. Condition 5 of the two definitions (for $\cap$) states that these presheaves should preserve pullbacks of monos; this is the critical property required for the sets-based presentation of this paper to work [17]. Presheaves enriched over types have appeared in [43], without the nominal sets style presentation and written for a different audience (one stemming from view of syntax and substitution based on [11]). The presheaves are used differently: by considering initial objects, inductive datatypes of *well-typed* syntax-with-binding are constructed.

Other theories of functions. *Combinatory algebra* (CA) assumes constants S, K, and I. Axioms allow them to model the λ-calculus. However, CA is strictly weaker than the theory of β-conversion; the (ξ) rule cannot be equationally axiomatised, because λ cannot be directly expressed (though any given λ-term can be compiled to combinators). This can be fixed using explicit indeterminates which, from the point of view of this paper, look a lot like atoms [37].

Alternatively, *lambda-abstraction algebras* (LAAs) are a first-order axiomatisation which does satisfy ξ [30]; again, from the point of view of this paper LAAs look much like the axioms we have considered. LAAs are not typed; a 'nominal' equivalent of them was considered by the first author with Mathijssen [23]. So this paper is significantly different from both since, as we see comparing this paper with [23], the addition of types makes a real difference to the models. LAAs take semantics in 'ordinary' sets, so their semantics is not well-pointed in the sense of this paper, and we do not obtain the language with meta-variables which we have developed here or relate so directly with a wider research context (e.g. into nominal techniques). One further subtle feature of LAA models is that they do not have finite support (we do not argue whether this is good or bad; we merely observe this as a significant difference).

Salibra and others have thought deeply about the lattice properties of λ-calculus models. As a final note we mention that nominal algebra satisfies an HSPA theorem [13], and *permissive*-nominal algebra satisfies an HSP theorem [15]. This has also been considered by Kurz and others [28]. The deeper theory here—how theorems of universal algebra applied to λ-calculus adapt to the nominal context—remains unexplored.

Other theories of existential variables. In implemented systems like LF and Isabelle [33] these are handled as a special syntactic category of higher-order variable. That is, an unknown of type ι depending on (universal) variables of type τ and τ' is modelled by a variable of type $\tau \to \tau' \to \iota$. We make no claim that our model of existential variables is better in implementation—it is simply too early to tell—but generally speaking we are against solving problems by moving to higher orders. Plenty of complexity can be *encoded* in function spaces, and this is fine for implementation, but encoding something is not the same as having a good mathematical model of it. Jojgov includes a excellent and detailed discussion of this issue in [27], which is his own analysis of incompleteness; intuitively, by conflating β-conversion with incompleteness it becomes impossible to distinguish between a complete derivation of higher type, and an incomplete derivation of lower type. We add that the denotation of $\tau \to \tau' \to \iota$ is uncountable, even if the denotations of τ, τ', and ι are countable. We would not immediately expect there to be uncountably many existential variables of type ι, so if only on the grounds of size we would hope for something smaller. Our denotations deliver this: an existential variable of type ι is just an unknown $X : \iota$.

Contextual modal type theory (CMTT) has two levels of variable; it enriches STLC with 'modal types' representing open code [31]. However, level 2 variables of CMTT are not existential variables; they are a species of *intensional* variable ranging over code. Making this formal using a nominal semantics related to the semantics of this paper, is current research by the first author.

6.2 Future work

We note that, as it stands, there is no general mathematical framework for the study of incomplete terms in type theory. Implementors of proof assistants invent *ad hoc* methods for representing incomplete terms, representing incomplete proof states, in their systems. Methods evolve more through trial and error than deep mathematical insight. For instance, early versions of Coq used a complex system involving two syntactic classes—'existential variables' and 'metavariables'—for representing incomplete proof states. Matita [1], whose design was influenced by lessons learned in Coq's development, used from the outset a much simpler scheme where the concepts of 'existential variable' and 'metavariable' are unified [4].

We hope that the work presented in Section 4 forms the basis for further, mathematical study of incomplete terms in type theory. For instance, the model of STLC in Definition 3.2 could be extended to a dependent type theory like e.g. compact λP [41, Subsection 14.2, Figure 14.1] (with or without incompleteness).

It should be fairly easy to internalise the substitution action for unknowns by adding λX, thus obtaining a two-level system with logic and computation at both levels—the result should resemble the first two levels of the lambda-context calculus [18], but with a stronger theory of α-equivalence and more reductions. One concrete application of this may be to expressing tactics—functions from incomplete derivations to incomplete derivations—in type-theory based theorem-provers, which need to program on terms (considered as computation or proof respectively). More goes into such a design than metavariables, but the character of metavariables is key to that of existing implementations [34].

Notions of incompleteness can be motivated by efficiency and speed; notably [35] was motivated by optimising unification in LF. These ideas have led to several implementations; an up-to-date overview is in [34]. Note that the details of the syntax are different: the work uses a two-level type system with special types for closed code, and 'meta-variables' range over *closed* elements of the domain (i.e. $supp(\varsigma(X)) = \varnothing$, intuitively). No general semantic theory has been given for this line of research, and we suspect that the nominal denotations of this paper could be turned to that task.

References

[1] Andrea Asperti, Claudo Sacerdoti Coen, Enrico Tassi & Stefano Zacchiroli (2007): *Crafting a Proof Assistant*, pp. 18–32. Lecture Notes in Computer Science, Springer, doi:10.1007/978-3-540-74464-1_2.

[2] Christoph Benzmüller, Chad E. Brown & Michael Kohlhase (2004): *Higher-order semantics and extensionality*. Journal of Symbolic Logic 69, pp. 1027–1088, doi:10.2178/jsl/1102022211.

[3] James Cheney & Christian Urban (2004): *Alpha-Prolog: A Logic Programming Language with Names, Binding and Alpha-Equivalence*. In Bart Demoen & Vladimir Lifschitz, editors: Proceedings of the 20th International Conference on Logic Programming (ICLP 2004), Lecture Notes in Computer Science 3132, Springer, pp. 269–283, doi:10.1007/978-3-540-27775-0_19.

[4] Claudio Sacerdoti Coen (2004): *Mathematical Knowledge Management and Interactive Theorem Proving*. Ph.D. thesis, University of Bologna.

[5] Gilles Dowek & Murdoch J. Gabbay (2010): *Permissive Nominal Logic*. In: Proceedings of the 12th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming (PPDP 2010), pp. 165–176, doi:10.1145/1836089.1836111.

[6] Gilles Dowek, Murdoch J. Gabbay & Dominic P. Mulligan (2009): *Permissive Nominal Terms and their Unification*. In: Proceedings of the 24th Italian Conference on Computational Logic (CILC'09).

[7] Gilles Dowek, Murdoch J. Gabbay & Dominic P. Mulligan (2010): *Permissive Nominal Terms and their Unification: an infinite, co-infinite approach to nominal techniques (journal version)*. Logic Journal of the IGPL 18(6), pp. 769–822, doi:10.1093/jigpal/jzq006.

[8] Maribel Fernández & Murdoch J. Gabbay (2007): *Curry-style types for nominal terms*. In: Types for Proofs and Programs (TYPES 06), Lecture Notes in Computer Science 4502, Springer, pp. 125–139, doi:10.1007/978-3-540-74464-1_9.

[9] Maribel Fernández & Murdoch J. Gabbay (2007): *Nominal rewriting (journal version)*. Information and Computation 205(6), pp. 917–965, doi:10.1016/j.ic.2006.12.002.

[10] Maribel Fernández & Murdoch J. Gabbay (2010): *Closed nominal rewriting and efficiently computable nominal algebra equality*. In: Electronic Proceedings in Theoretical Computer Science, 34, pp. 37–51, doi:10.4204/EPTCS.34.5.

[11] Marcelo P. Fiore, Gordon D. Plotkin & Daniele Turi (1999): *Abstract Syntax and Variable Binding*. In: Proceedings of the 14th IEEE Symposium on Logic in Computer Science (LICS 1999), IEEE Computer Society Press, pp. 193–202, doi:10.1109/LICS.1999.782615.

[12] Murdoch J. Gabbay (2007): *Fresh Logic*. Journal of Applied Logic 5(2), pp. 356–387, doi:10.1016/j.jal.2005.10.012.

[13] Murdoch J. Gabbay (2009): *Nominal Algebra and the HSP Theorem*. Journal of Logic and Computation 19(2), pp. 341–367, doi:10.1093/logcom/exn055.

[14] Murdoch J. Gabbay (2011): *Foundations of nominal techniques: logic and semantics of variables in abstract syntax*. Bulletin of Symbolic Logic 17(2), pp. 161–229, doi:10.2178/bsl/1305810911.

[15] Murdoch J. Gabbay (2011): *Nominal terms and nominal logics: from foundations to meta-mathematics*. In: Handbook of Philosophical Logic, 17, Kluwer.

[16] Murdoch J. Gabbay (2011): *Two-level nominal sets and semantic nominal terms: an extension of nominal set theory for handling meta-variables*. Mathematical Structures in Computer Science doi:10.1017/S0960129511000272. Published online.

[17] Murdoch J. Gabbay & Martin Hofmann (2008): *Nominal renaming sets*. In: Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2008), Springer, pp. 158–173, doi:10.1007/978-3-540-89439-1_11.

[18] Murdoch J. Gabbay & Stéphane Lengrand (2009): *The lambda-context calculus (extended version)*. Information and computation 207, pp. 1369–1400, doi:10.1016/j.ic.2009.06.004.

[19] Murdoch J. Gabbay & Aad Mathijssen (2006): *One-and-a-halfth-order logic*. In: Proceedings of the 8th ACM-SIGPLAN International Symposium on Principles and Practice of Declarative Programming (PPDP 2006), ACM, pp. 189–200, doi:10.1145/1140335.1140359.

[20] Murdoch J. Gabbay & Aad Mathijssen (2008): *Capture-Avoiding Substitution as a Nominal Algebra*. Formal Aspects of Computing 20(4-5), pp. 451–479, doi:10.1007/11921240_14.

[21] Murdoch J. Gabbay & Aad Mathijssen (2008): *One-and-a-halfth-order Logic*. Journal of Logic and Computation 18(4), pp. 521–562, doi:10.1093/logcom/exm064.

[22] Murdoch J. Gabbay & Aad Mathijssen (2009): *Nominal universal algebra: equational logic with names and binding*. Journal of Logic and Computation 19(6), pp. 1455–1508, doi:10.1093/logcom/exp033.

[23] Murdoch J. Gabbay & Aad Mathijssen (2010): *A nominal axiomatisation of the lambda-calculus*. Journal of Logic and Computation 20(2), pp. 501–531, doi:10.1093/logcom/exp049.

[24] Murdoch J. Gabbay & Andrew M. Pitts (2001): *A New Approach to Abstract Syntax with Variable Binding*.

Formal Aspects of Computing 13(3–5), pp. 341–363, doi:10.1007/s001650200016.

[25] Jean van Heijenoort (1967): *From Frege to Gödel: a source book in mathematical logic, 1879-1931*. Harvard University Press.

[26] Leon Henkin (1950): *Completeness in the Theory of Types*. *Journal of Symbolic Logic* 15, pp. 81–91.

[27] Gueorgui I. Jojgov (2002): *Holes with Binding Power*. In: *TYPES, Lecture Notes in Computer Science* 2646, Springer, pp. 162–181, doi:doi:10.1007/3-540-39185-1_10.

[28] Alexander Kurz & Daniela Petrişan (2010): *On Universal Algebra over Nominal Sets*. *Mathematical Structures in Computer Science* 20, pp. 285–318, doi:10.1017/S0960129509990399.

[29] Karel Lambert (1963): *Existential Import Revisited*. *Notre Dame Journal of Formal Logic* 4(4), pp. 288–292.

[30] Giulio Manzonetto & Antonino Salibra (2010): *Applying Universal Algebra to Lambda Calculus*. *Journal of Logic and computation* 20(4), pp. 877–915, doi:10.1093/logcom/exn085.

[31] Aleksandar Nanevski, Frank Pfenning & Brigitte Pientka (2008): *Contextual modal type theory*. *ACM Transactions on Computational Logic (TOCL)* 9(3), pp. 1–49, doi:10.1145/1352582.1352591.

[32] John von Neumann (1929): *Über eine Widerspruchsfreiheitsfrage in der axiomatischen Mengenlehre*. *Journal für die reine und angewandte Mathematik* 160.

[33] Lawrence C. Paulson (1989): *The Foundation of a Generic Theorem Prover*. *Journal of Automated Reasoning* 5(3), pp. 363–397, doi:10.1007/BF00248324.

[34] Brigitte Pientka & Joshua Dunfield (2010): *Beluga: A Framework for Programming and Reasoning with Deductive Systems (System Description)*. In: *Proceedings of the 5th International Joint Conference on Automated Reasoning (IJCAR 2010), Lecture Notes in Computer Science* 6173, Springer, pp. 15–21.

[35] Brigitte Pientka & Frank Pfenning (2003): *Optimizing Higher-Order Pattern Unification*. In: *Proceedings of the 19th International Conference on Automated Deduction (CADE 2003), Lecture Notes in Computer Science* 2741, Springer, pp. 473–487.

[36] Bertrand Russell (1905): *On Denoting*. *Mind, New Series* 14(56), p. 479493.

[37] Peter Selinger (2002): *The lambda calculus is algebraic*. *Journal of Functional Programming* 12(6), pp. 549–566, doi:10.1017/S0956796801004294.

[38] Mark R. Shinwell, Andrew M. Pitts & Murdoch J. Gabbay (2003): *FreshML: Programming with Binders Made Simple*. In: *Proceedings of the 8th ACM SIGPLAN International Conference on Functional Programming (ICFP 2003)*, 38, ACM Press, pp. 263–274, doi:10.1145/944705.944729.

[39] Raymond Smullyan (1968): *First-order logic*. Springer. Reprinted by Dover, 1995.

[40] Christian Urban, Andrew M. Pitts & Murdoch J. Gabbay (2004): *Nominal Unification*. *Theoretical Computer Science* 323(1–3), pp. 473–497, doi:10.1016/j.tcs.2004.06.016.

[41] Pawel Urzyczyn & Morten Sørensen (2006): *Lectures on the Curry-Howard isomorphism*. *Studies in Logic* 149, Elsevier.

[42] Claus-Peter Wirth (2004): *Descente Infinie + Deduction*. *Logic Journal of the IGPL* 12(1), pp. 1–96, doi:10.1093/jigpal/12.1.1.

[43] Julianna Zsidó (2010): *Typed abstract syntax*. Ph.D. thesis, University of Nice.